Constructive solid geometry with projection.

Tongsiri, Natee

Award date:
2001

Awarding institution:
University of Bath

Constructive Solid Geometry with Projection

submitted by

Natee Tongsiri

for the degree of Doctor of Philosophy

of the

University of Bath

2001

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Acknowledgments

I would like to thank:

My super supervisor Dr Daniel Richardson for his ideas, support, guidance and encouragement throughout the research.

The Geometric Modelling Group, University of Bath. Dr Adrian Bowyer, Dr Kevin Wise, Dr David Eisenthal, Dr Jakob Berchtold, Dr Dan Pidcock and Dr Irina Voiculescu.

Members of the Department of Computer Science. Most notably, Professor John ffitch for the latest REDUCE, Dr Nicolai Vorobjov, Dr Julian Padget and Dr Russell Bradford for their comments and suggestions.

My family for their love and support.

My sponsor, The Royal Thai Government and the Department of Mathematics, Chiang Mai University, for the financial support.

Postgraduate students and research officers of the Computing Group for keeping me distracted from my work. Particularly, Adam Batenin, Dr William Naylor, Dr David Power, Dr John McDermott, Dr Nicholas Howgrave-Graham, Joy Lu, Dr Nam Hur and Andy Holt.

Westwood Resident Tutor Team for their support and understanding through out my Resident Tutor career. In particular, I would like to thank Anne Wright for the exclusive use of her kitchen.

Douglas Adams. So long, and thanks for all the books.

Summary

We think the Configuration Space approach to spatial planning problem is good. The configuration space obstacles are geometric objects that can be represent using semi-algebraic CSG representation. Moreover, they can also be represent using existential quantifiers which correspond to geometric projections. If the projected variables only occur algebraically then it is possible to eliminate quantifiers and represent configuration space obstacles in the extended semi-algebraic form. We think this computation can be done more efficiently if it is preceded by spatial subdivision and pruning. However, no matter how it is done the quantifier elimination is computationally hard, and the output in extended semi-algebraic representation is large and cumbersome.

It seems that we should learn to work directly with the representation of the configuration space obstacle as a projection in an extended CSG system. In any case, if the moving object or part of the obstacle is not algebraic, we must represent the C-space obstacle as a projection since elimination of quantifiers may not be possible. Therefore, we are looking into the use of an extended CSG system which has bounded projection and boundary formation as operators, as well as the usual Boolean ones.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

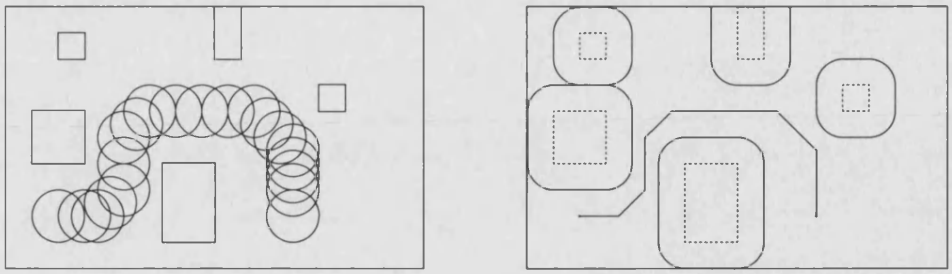
1.1 Thesis Overview

In this thesis we present methods for computing constraints on the position and orientation of an object due to the presence of other objects. These constraints problems arise in applications that require choosing how to arrange objects or how to move objects without collisions. We refer to this type of problem as a *Spatial Planning* problem.

One major approach to solve spatial planning problems is to use the concept of *Configuration Space* introduced by Lozano-Pérez [41]. This approach simplifies the problem from having to deal with the intersections between set of objects to dealing with a point relative to a set of objects instead. The configuration forbidden to an object due to the presence of other objects can be characterised as regions in a *configuration space* which we refer to as *configuration space obstacles.*

We demonstrate the idea of configuration space with the following example.

Consider trying to guide a fat man who was lost in a forest. Assume that the cross-section of this man is uniformly circular, whereas the cross-sections of some trees are square and some are rectangular in shape. The view from the sky above



Workspace C-space

Figure 1-1: Fat man in the forest; Workspace and Configuration space.

the forest is the view of our *Workspace* (Figure 1-1).

One way of solving the problem is to find the route for the man by measuring all gaps between the trees and identify the gaps that are bigger than his diameter. Alternatively, we can imagine that the man has lost weight so that his cross-section has shrunk to a point and all the trees have grown by the size of the radius of the man. Looking at the problem this way, we only have to guide a single point around the 'grown' obstacles.

We refer to this somewhat transformed Workspace as the *configuration space* or C-space. The point which represents the man is called the *reference point.* All grown trees are called *configuration space obstacles.* Each point in the configuration space corresponds to a position of the man in the Workspace relative to his reference point. Every point in the configuration space obstacle corresponds to a position in the Workspace where the man comes into contact with the tree.

In this thesis we consider the case of a single 2-dimensional object moving in the presence of several 2-dimensional static obstacles in a 2-dimensional Workspace. We allow the moving object and the obstacles to have curved edges and they may also be non-convex. Additionally, we allow both translational and rotational movements of the moving object, thus the moving object has 3 *degrees of freedom.*

In order to be able to represent a position and orientation of the actual object in the Workspace by a single point in the configuration space, we represent the rotational movement of the object by adding an extra dimension to the config-

uration space. In general, at least n dimensions is needed for the configuration space so that each point in this configuration space represents a configuration of an object which has n degrees of freedom.

Computing the C-space obstacles is an important process in the C-space approach to spatial planning. *We think a good approach to represent the constraints of an object due to the presence of other objects is by constructing a configuration space obstacle using the idea of Constructive Solid Geometry (CSG) and semi-algebraic sets.*

Model in CSG are built up from a Boolean combination of primitive geometric objects such as half-planes, cylinders and spheres. We call these base-objects *primitives.* Since an arbitrarily complicated geometric shape has semi-algebraic description, we can define most CSG primitives by using semi-algebraic sets which are subsets of some R^n defined by a finite number of polynomial equations and inequalities [4]. Consequently, we can use these sets to represent n-dimensional object in n-dimensional space.

Additionally, semi-algebraic CSG representation allows the processes of pruning and subdivisions to be done on geometric objects [69]. Complicated CSG objects can be represented by a tree structure with Boolean operators on the internal nodes, and primitives at the leaves. To answer a query about an object represented in this way is computationally expensive since every node and leaf has to be consulted. One way to overcome this problem is to restrict the number of nodes and leaves that need to be considered. We can impose these restrictions by dividing the space that contains the tree into smaller spaces, and by pruning the tree for each space. Based on the assumption that 'representation of objects may be globally complicated but locally simple', pruning and subdivisions can lead to simpler representations of objects. Each time the subdivision occurs, the smaller sub-space may have simpler objects while the combination of all spaces still represent the more complicated original model.

Moreover, not only that semi-algebraic sets are closed under finite union, intersection and negation, they are also closed under elimination of quantifier. A quantifier is denoted by one of the two symbols; $\forall$ (for all) and $\exists$ (there exist).

We use quantifiers in the form, for example; $(\forall x)p(x)$ which means 'for all x, the expression $p(x)$ is true', or $(\exists x)p(x)$ which means 'there exists x such that the expression $p(x)$ is true'. Given a logical expression containing quantifiers, the problem of quantifier elimination is that of finding an equivalent expression which does not contain any quantifiers.

Existential quantifiers correspond to geometric projections. We can regard a set defined by $B = (\exists x)A$ as a set of points at which A is true. The set of points defined by B is a projection of a set defined by A. The projection is parallel to the x axis onto the space of other variables of A. In theory, if the CSG primitives are semi-algebraic then the quantifier can be eliminated and the results are also semi-algebraic.

In addition to existential quantifier, boundary of semi-algebraic sets are also semi-algebraic. Thus we can also represent C-space obstacles using a combination of existential quantifiers and boundary of semi-algebraic sets. We demonstrate that the configuration space obstacle is naturally a projection of a higher-dimension object onto the configuration space. Therefore existential quantifiers which correspond to geometric projections can be used to represent C-space obstacles.

In order to represent semi-algebraic CSG objects with Boolean operators alone, one approach is to apply elimination of quantifiers. Although possible in theory, the process of quantifier elimination is computationally expensive and quantifier-free results are often large and cumbersome and may not obtainable in a reasonable amount of time.

Since the process of quantifier elimination has computational difficulty which increases much more than linearly with complexity of formula, *we explore the possibility of using spatial subdivision techniques as a pre-process before applying quantifier elimination.*

We can also extend the CSG primitives beyond the semi-algebraic. It is often useful in applications to consider trigonometric and exponential functions. We can extend the semi-algebraic sets to include these functions. We refer to these sets as *Extended Semi-algebraic sets.* Extended semi-algebraic sets have good

expressive power. Not only that they are capable of representing static objects, it is also natural to describe motion constraints of objects using these sets.

However, the subsets of R^n which can be represented by Boolean combinations of the extended semi-algebraic sets as primitives do not have such good closure properties as the semi-algebraic set primitives. The projections of extended semi-algebraic sets are only guaranteed to be extended semi-algebraic if the variables being projected only occur algebraically. In this thesis, we restricted ourselves to these cases.

Another approach is to extend the set of operators to include projection and boundary formation as well as the usual set operators. This allows a more compact representation of C-space obstacles but also raises new difficulties. We do not yet fully understand how to compute with sets define with these new operators. *We discuss how to compute with such extended system. We also investigate how the quantifier elimination process might be combined with pruning and recursive subdivision and extended to deal with the two new operators.*

We also explore the possibility of applying the extended CSG system to spatial planning. The C-space obstacle, which is the solution to Find-space problem, can be represented using extended semi-algebraic sets with extended operators.

1.2 Terms and Definitions

In many occasions we use the term *object* instead of 'moving object' to refer to the movable rigid-body. The term *obstacle* refers to a static n-dimensional rigid-body whereas *obstacles* refer to a group of obstacles which may or may not be connected.

We also adopt some general standard notations and meanings. All geometric entities – lines, edges, faces and objects will be treated as infinite sets of points. All of these entities will be in some R^n, an n-dimensional Euclidean space. We denote points of R^n by a, b, x, y, z and denote sets of points in R^n by A, B, C, P

and O. Additionally, we use i, j, k, l to denote integers.

1.3 Thesis Outline

This thesis is organised around 7 chapters which are outlined below.

In this Chapter, we give an overview of the thesis and define some key terms which will be used throughout.

Chapters 2-4 provides the background knowledge to this thesis. In Chapter 2 we describe the concept of spatial planning problem and configuration space. We also outline the configuration space approach to spatial planning by means of constructing configuration space obstacles. In Chapter 3 we provide a brief description of CSG and a definition of semi-algebraic sets. We also summarise the mathematical framework of the semi-algebraic approach to CSG. We then give an overview of Svlis geometric CSG modeller and describe in details the pruning and recursive subdivision technique. We also mention some existing semi-algebraic CSG representations of configuration space obstacles. In Chapter 4 we briefly describe the quantifier elimination method. We also explain how it can be used to construct configuration space obstacles.

Chapter 5 and Chapter 6 contains the main part of the work. In Chapter 5 we define the C-space obstacles of a 2-dimensional object translating and rotating freely among 2-dimensional obstacles using existential quantifiers. We then provide simple algorithms to compute these C-space obstacles using the quantifier elimination technique of Cylindrical Algebraic Decomposition. We also suggest using pruning and subdivision process to simplify the problem before applying quantifier elimination. We then give details of the experiment and report the result. In Chapter 6 we give precise definition of Extended CSG System. We also outline how the pruning and recursive subdivision technique to this extended system. We describe in detail how we may construct configuration space obstacles using the two new operators of our extended system. We also investigate the use of pruning and subdivisions to conservatively generate Freespace and

demonstrate the method of finding a partial solution to the Find-path problem.

Finally, in Chapter 7 we summarise the work of this thesis, highlight the original concepts which we introduced and outline some problems which remain unsolved.

Chapter 2

C-space Approach to Spatial Planning

In this chapter we briefly describe the problem of spatial planning and describe in detail the concept of configuration space. We also give a brief survey of the configuration space approach to spatial planning.

2.1 Spatial Planning

Spatial planning problems are a class of geometric problems which involve placing an object among other objects or moving an object from one place to another without colliding with other objects in the process.

We refer to the problem of placing an object among obstacles as a *Find-space* problem and refer to the problem of finding a collision-free path for an object as a *Find-path* problem.

Let A be an object and B_j be a set of, possibly intersecting, objects. Let R be an object that completely contains A and B_j. Find-space and Find-path can be defined as follows.

1. *Find-space* – Find a position for A inside R, such that for all B_j, $A \cap B_j = \phi$.

2. *Find-path* – Find a path for A from position p_0 to position p_1 such that A is always in R and on this path $A \cap B_j = \phi$.

Spatial planning is a computationally difficult problem. Not only that the complexity of the computation increases with the number of dimensions and the number of objects involved, it also depends on the representation and the complexity of objects. Much research has been devoted to the complexity aspect of the spatial planning problem. For example, it was studied by Canny [11], Davenport [14], Hopcroft et al. [32], Lozano-Pérez [41], Reif [54] [55], Schwartz and Sharir [57] [58], Vanderstappen et al. [66].

2.2 C-space

In 1983, Lozano-Pérez [41] introduced the formal idea of configuration space to spatial planning. The main idea of this approach is to map the original problem from lower dimension to a relatively simpler problem in higher dimension.

For example, consider a typical packing problem where the original problem involves the optimal orientation of identically-shaped polygons. We can reduce the problem of how to minimally pack polygons in a thin rectangular space, so that each polygon has the same orientation, to the case of finding the minimal width of the cross section of a torus defined by quadratic surface patches instead [54].

Based on the assumption that a set of parameters can represent the configuration of a solid object in space, this formalised approach represents a position and orientation of an object as a single point in the space of these parameters.

For example, consider a polygon which may rotate and translate freely in 2-dimensional space. We can represent its configuration as a point in 3-dimensional space using three parameters where 2 parameters correspond to each dimension

of its translations and one parameter correspond to its rotation. Similarly, we can represent a configuration of a polyhedron which may rotate and translate freely in space as a point in 6-dimensional space using six parameters where 3 parameters correspond to each dimension of the translations and 3 parameters correspond to each dimension of the rotations.

A point in this created parameters space corresponds to a specific position and orientation of the actual object in the original space. Consequently, this parameters space is sufficient to represent every conceivable position of the actual object in the original space. We call the space of these parameters the *Configuration Space* or *C-space* and we refer to the original space as the *Workspace.*

2.2.1 Dimensions of C-space

We define the number of dimensions in which a particular object can 'move' in space to be its *degrees of freedom.* Since a 'configuration' of a particular object refers to both its position and its orientation, the degrees of freedom of an object, which allow to translate and rotate, is a combination of degrees of freedom of its translational and rotational movement.

The maximum degrees of freedom of an object depend on the dimensions of its Workspace. Consequently, the dimension of the Workspace dictates the dimension of the C-space. An object in n-dimensional Workspace has at most $n + (\frac{1}{2}n(n-1))$ degrees of freedom, where n-dimensions correspond to the translations and $(\frac{1}{2}n(n-1))$ correspond to the rotations [6].

In general, the minimum number of independent parameters required to specify every conceivable position and orientation of an object, relative to a frame of reference, is equal to the number of its degrees of freedom. For example, a configuration of an object in n-dimensional Workspace may be regarded as a point in R^d where $d = n + (\frac{1}{2}n(n-1))$. In other words, a configuration of an n-dimensional object which has d degree of freedom can be specified using d parameters. Thus, in general, the minimum dimensions of a C-space is equal to

the degrees of freedom of the object.

However, one may wish to represent the three degrees of freedom for rotational movement using quaternions which have 4 variables. Quaternions and their application in C-space were discussed in [11] [38] [39].

2.2.2 Reference Point

Define the *reference point* of an object as a fixed point in the Workspace coinciding with the origin of the global coordinate frame. Denote the reference point of an object A by rp_A. The configuration of an object is defined in terms of its reference point, relative to its initial configuration, by a number of parameters corresponding to the degrees of freedom of the object. We denote the initial configuration of A in terms of rp_A by A_0 and denote the position of A in the configuration α by A_α.

Generally, an object is placed in the coordinate frame in such a way that the reference point is inside the object. However, the reference point does not have to be inside the object. It can be a point outside the object as illustrated in Figure 2-1.

By convention, a position of a translated object in the Workspace is specified by its reference point relative to its coordinate frame. A *position* is the distance from the origin of the coordinate frame to the reference point of the object in each translational dimension. In contrast, the *orientation* of the object is specified relative to the original orientation of the object itself. Generally, the orientation of an object is the anti-clockwise angle about the reference point of the object in each of the rotational dimension.

Figure 2-2 illustrates the C-space of a 2-dimensional object A in R^2 with the translation of reference point rp_A relative to the origin of the coordinate frame, and a rotation around rp_A. The configuration of A can be specified by three parameters (z, w, θ) where (z, w) is the position of rp_A and θ is the angle of rotation about rp_A relative to A_0.

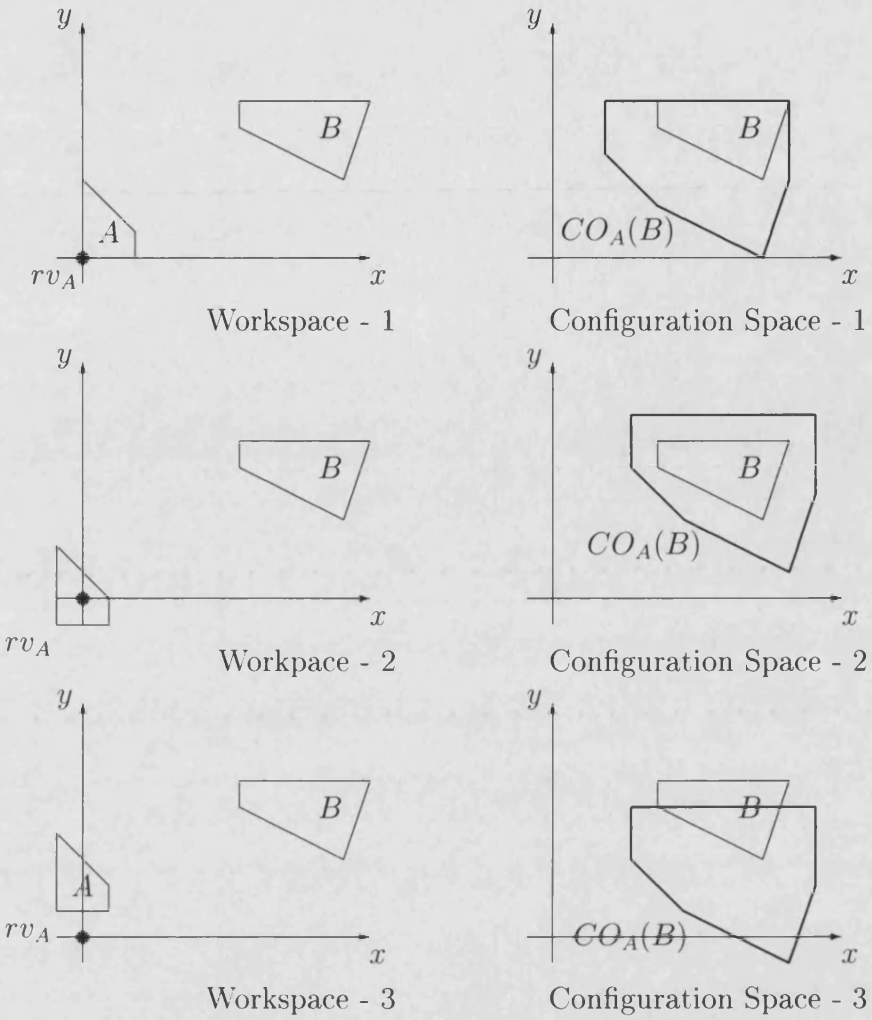


Figure 2-1: The choice of a different reference point result in a different C-space obstacle.

The C- space of A, denoted by $Cspace_A$, can be regarded as the space R^3. However, the space $R^2 \times [0 : 2\pi)$ suffices to represent the C-space of 2-dimensional moving object since the point $(z, w, 0)$ corresponds to the object in the same configuration in the Workspace as the point $(z, w, 2\pi)$ [17]. Similarly, if the orientation of A is fixed then R^2 is enough to specify the configuration of A.

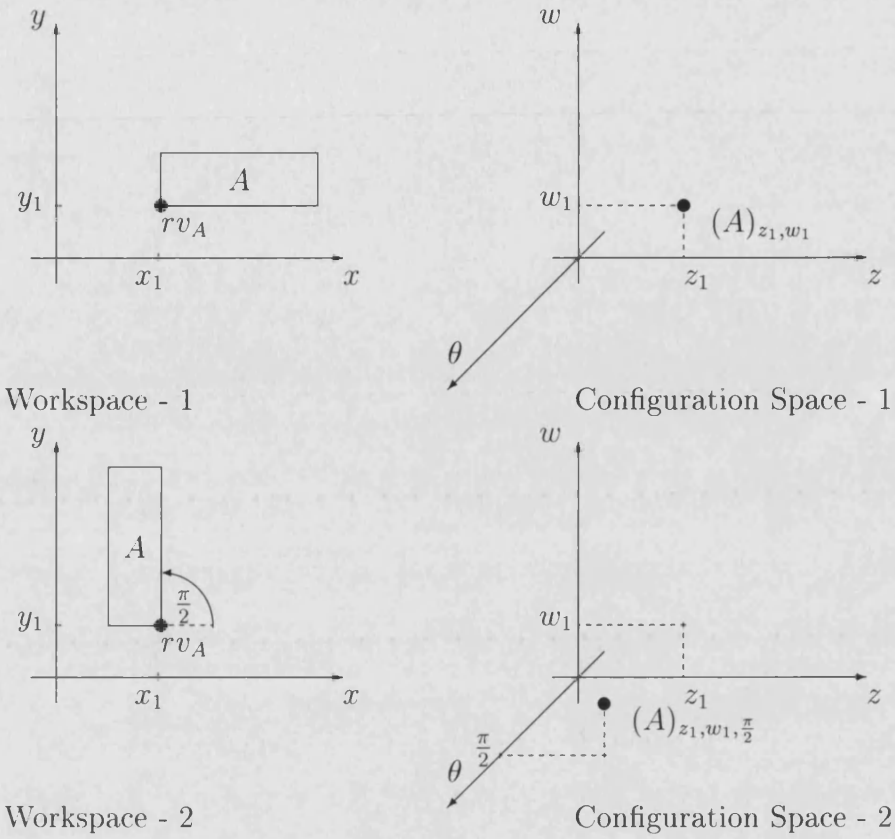


Figure 2-2: A configuration of a polygon which may translate and rotate can be specified by three parameters – 2 parameters correspond to the two dimensions of the translation and one parameter corresponds to the rotation.

2.2.3 C-space Obstacle

An object and an obstacle cannot occupy the same point in the Workspace therefore not all the points in the C-space are valid. Some points in the C-space may correspond to a configuration of the object in the Workspace where it intersects the obstacle. *C-space obstacle* is the collection of such points.

For example, consider an object which is only a point in space. If the reference point is inside the object, i.e. the reference point is the object itself, then the obstacle in the Workspace is identical to the C-space obstacle.

We define C-space obstacles as follows:

Definition 1 *Define $Cspace_A$ obstacles due to B, denoted by $CO_A(B)$ as:*

$$CO_A(B) \equiv \{x \in Cspace_A \mid A_x \cap B \neq \phi\}$$

All points in the C-space which correspond to all the points where the object do not intersect any obstacle is called *Freespace* and the boundary between C-space obstacle and Freespace is referred to as the *contact surface.*

The choice of a different reference point results in a different C-space obstacle. Figure 2-1 illustrates the case of a 2-dimensional object A in R^2, which may translate but not rotate. The constraint on the configuration of A due to the obstacle B is all the positions of rp_A outside $CO_A(B)$. A different position of rp_A results in the translation of $CO_A(B)$ relative to the obstacle B.

Lozano-Pérez [41] demonstrated that if an object is completely enclosed in another object then the C-space obstacle of the larger object alone suffices to represent both object in C-space. Additionally,

$$CO_A(B_1 \cup B_2) \equiv CO_A(B_1) \cup CO_A(B_2).$$

Minkowski Sums

The idea of C-space obstacles is closely related to that of Minkowski sums of sets.

Define *Minkowski sums* as:

$$A \oplus B = \{a + b \mid a \in A, b \in B\}$$

where $A, B \subset R^2$ and $a + b$ denotes the vector sums of the vectors a and b.

If $a = (a_i, \ldots, a_n)$ and $b = (b_i, \ldots, b_j)$ then

$$a + b = (a_i + b_i, \ldots, b_n + b_n)$$

and for a set A define

$$-A = \{-a : a \in A\}$$

The C-space obstacles can be expressed as Minkowski sums as follow.

Theorem 1 *Let A be a translating polygon and let B be an obstacle. The $Cspace_A$ due to obstacle B or $CO_A(B)$ is:*

$$\{(x, y) : (x, y) \in B \oplus -A_0\}.$$

This theorem, which was proved by Lozano-Pérez [41], extends to higher dimension as long as the orientation of A is fixed. Also, if A and B are convex polygons with m and n edges respectively, the Minkowski sums $A \oplus B$ is a convex polygon with at most $m + n$ edges. However, the complexity of the Minkowski sum increase to $O(mn)$ when one of the polygon is non-convex and become $O(m^2n^2)$ when both polygons are non-convex [17]. Moreover, In the case of a convex polyhedron translating in 3-dimensional space amidst k convex polyhedral obstacles, Aronov and Sharir demonstrated in [1] that the Freespace which is the complement of the union of the Minkowski sums has combinatorial complexity of $O(nk \log k)$, where n is the total complexity of each k Minkowski sums.

Minkowski sums are also closely related to an important geometric notion of convolution which is the computation of the sweep volume of an object moving along a trajectory. Convolution can also be represented as a C-space obstacle as shown by Kim in [37].

Although the idea of Minkowski sums does not extend directly to deal with rotating object, it give us insight to what C-space obstacle of object which may rotate may look like.

Sweeping Minkowski Sum

Consider a 2-dimensional object which may translate and rotate avoiding some obstacles in xy-space. Denote the translation in x and y dimension by z and w respectively. Also, denote the angle of the object when rotating around its reference point by θ. The C-space obstacle of this object can be regarded a 3-dimensional object in $zw\theta$-space.

Each cross-section of this object along θ axis correspond to the translational C-space obstacle of this object at a certain θ. On this plane, we are dealing only with a translational problem of a moving the object at an angle θ to its original orientation. Minkowski sums can be used to find the C-obstacles at this θ.

Theorem 2 *Let A be a polygon which can translate and rotate. Let B be an obstacle. The $Cspace_A$ due to obstacle B or $CO_A(B)$ is:*

$$\{(x, y, \theta) : (x, y) \in B \oplus -A_0(\theta)\}.$$

A consequence of this is that we can find C-space obstacles with geometric operations of rotation and Minkowski sum. Regarding θ as a variable, we get a representation of the C-space obstacle in (x, y, θ)-space. However, the obstacle is no longer linear.

2.3 C-space Approach to Spatial Planning

Since C-space was formalised, a great deal of spatial planning research using C-space approach has been carried out. Practical algorithms for many special cases have been implemented. For example, an efficient algorithm known for generating C-space obstacles have been only for polyhedral object and obstacles using method of computing convex hull and Minkowski sums [41]. Many more were described by Latombe [40]. However, most algorithms are for polygons

and polyhedra only since they rely on certain properties of polygons contact conditions.

2.3.1 Characteristics of C-space Approach

A few prominent characteristics of C-space approach to spatial planning include:

Prior Knowledge Complete prior knowledge of the environment needs to be provided in order to do the calculation. The case of an unknown environment was studied by, for example Chien et al. [12], Lumelsky [43], Skewis and Lumelsky [62].

Object Solidity and Rigidity To ensure that their configuration can be parameterised with a few parameters and that they retain the shape while moving. The case of flexible object was studied by, for example Hopcroft et al.[31].

Static Environment The obstacles may not rotate or translate. The case of dynamic environment was studied by, for example Fujimura [24] [25], Lumelsky [42], Pan and Luo [47].

Arbitrary Directions Objects are treated as a free-floating bodies in space. The calculation does not take into account the constraints caused by the mechanics of the object such as, the turning circle of vehicles with wheels. The case of car-like robot was studied by, for example Bicchi et al. [5], Desaulniers and Soumis [18].

2.3.2 C-space Representations

An important aspect of spatial planning systems that use C-space approach is the method of representing the C-space. The representation method needs to be able to classify regions of C-space into, at least, 2 sub-regions; corresponding to

where the object in the Workspace can and cannot go, and enable the search for positions within these regions [9].

Many C-space representations exist, each has advantages and disadvantages over the other. Recent surveys, for example, by Hwang [35], Wise [67] [68], described many techniques used to compute and represent C-space which can be classified into many sub-categories, in many different ways. For example, the computation methods can be differentiate between global to localised approach, numerical as opposed to analytical computation, or approximate as opposed to exact computation.

One of the major technique of representing C-space is cell decomposition which was adopted by, for example, Brooks et al. [10], Faverjon [23], Lozano-Pérez [41], Schwartz and Sharir [57] [58] [59] [60], Sharir and Sheffi [61]. The technique is based on discretising the C-space into a finite number of cells and use some tests to classified each cells whether it belongs to the Freespace or the C-space obstacle. By building a connectivity graph which represent adjacency relation of these cells, path planning become a graph-search problem in which many efficient algorithms exist [33]. Additionally, potential field techniques can be used in conjunction with the connectivity graphs for path planning. Localised potential field technique was studied by, for example Barraquand and Latombe [2] [3] where the example of globalised version was studied by Hwang and Ahuja [34].

2.4 Summary

In this chapter we described many background ideas. We gave the definition of spatial planning and explained the idea of configuration space. We also mentioned many previous works on configuration space approach to spatial planning which gave us many insights into the nature and magnitude of the problem.

Chapter 3

Constructive Solid Geometry

In this chapter we give a brief description of Constructive Solid Geometry (CSG). We also summarise the mathematical framework of the semi-algebraic approach to CSG. In Section 3.3, we give an overview of a kernel geometric modeller – Svlis, which uses semi-algebraic CSG representation to represent its models. We also outline the technique of pruning and recursive subdivision, employed by Svlis, in Section 3.4. In the last section we identify several semi-algebraic CSG approaches to spatial planning.

3.1 Constructive Solid Geometry

Geometric models are artificially constructed geometric objects that make the investigation of the actual object easier. Models are useful because a study for certain characteristics can be carried out more easily on the model than on the object itself. Moreover, geometric models are not restricted to represent only real objects, they can also represent artificial objects.

Configuration space obstacle of the moving object among obstacles can be regarded as a geometric object. The solid part of the model could corresponds to the configuration that would cause the object to collide with the obstacles where

empty space outside the solid corresponds to the possible configuration of the object in the Workspace.

There are many methods available to represent geometric models. Some common methods include:

Edge List which uses the list of edges to describe a 2-dimensional object. It can also be used to model a wire-frame of a 3-dimensional object.

Boundary Representation which represents the surface of an object explicitly but represent the interior only implicitly. Objects are represented as a list of faces as planar polygons. Each polygons are represented by its vertices and edges. Other physical properties of the object may also be part of the representation. This is the most common method of solid modelling.

Bicubic Surface Patches which represents curved surface patches using cubic polynomial functions with two parameters.

Implicit Equation which describes curves and surfaces using implicit equations. For example, the equation

$$ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0$$

defines the family of quadric surfaces. This implicit equation can define, for example, spheres, cylinders, ellipsoids or paraboloids depending on the values or the parameters $a, \ldots, k$.

Sweep Representation which generates a model by sweeping a 2-dimensional cross-section along a curve. For example, a circle swept along a straight line generates a cylinder and a circle swept along another circle generates a torus. A useful technique is to vary the size of the cross-section as it sweep. For example, a circle sweeping along straight line with the radius linearly decreasing generates a cone.

CSG or Set-Theoretic Modelling which describes the geometry of a complex object by combining simple objects using operators of Set Theory. Complicated CSG objects can be treated as though they were a single object and

can be combined to make more complicated objects in the same manner. CSG method is useful both as a method for representing geometric object and as an intuitive user interface technique.

CSG is widely studied and CSG models are regarded as more stable than others because the properties of its Boolean operators are well understood [27]. Additionally, efficient $O(nlogn)$ algorithm to convert boundary representation to CSG representation exists for a simple polygon of n sides [19]. However, the algorithm does not extend to polyhedra.

Simple objects in CSG are referred to as *primitives*. In order to representing CSG primitives, we need to consider a computable representation. Semi-algebraic sets, which will be defined in the next section, appear to be an appropriate candidate since they have good expressive power. For example, primitives such as half-spaces, spheres and cylinders are easy to represent in semi-algebraic form. Additionally, semi-algebraic is a natural way to describe geometric constraints [11].

Gomes and Teixeira [27] described in details, the mathematical framework for computable CSG primitives using levels of decreasing abstraction from Boolean algebra of sets, set-point topology and geometry to semi-algebraic sets. The next section summarises this framework.

3.2 Mathematical Framework

Since we are interested in a class of geometric objects which can be represented and processed in a computer system, the representation of geometric objects needs to be computable. The computability of geometric objects depends on the fact that it presents finite describability of the objects and capable of algebraically combine existing objects in order to generate new ones. The Boolean algebra of semi-algebraic sets is such an algebraic structure which provides these properties.

3.2.1 Boolean Algebra

An algebraic structure is a set with operations denotes by a pair $\langle S; \Omega \rangle$ where S is a set and Ω is a collection of operations defined on S.

A Boolean algebra is an *abstract* algebraic structure in which only the general properties are described but the sets and the operations are not defined.

Definition 2 *A Boolean algebra with universe B is an algebraic structure $\langle B; +, \cdot, - \rangle$ with 0 and 1 as distinct elements of B; $+$, $\cdot$ are binary operations on B; and $-$, is a unary operation on B such that*

1. $a + b = b + a$
2. $a + (b + c) = (a + b) + c$
3. $a + (b \cdot c) = (a + b) \cdot (a + c)$
4. $0 + a = a$
5. $a + \overline{a} = 1$
6. $a \cdot b = b \cdot a$
7. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
8. $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
9. $1 \cdot a = a$
10. $a \cdot \overline{a} = 0$

The theory of Boolean algebra can be extended to the theory of sets. For example, a Boolean algebra which elements are sets and the operations on sets are union ($\cup$), intersection ($\cap$) and complement($\setminus$) : $\langle S; \cap, \cup, \setminus \rangle$, is called a Boolean algebra of sets.

The above example of an algebraic structure is said to be at a *concrete* level and is the level of abstraction concerning geometric modelling. Geometric modelling always require to know the element of the set S and that there are rules to evaluate the operations of Ω over S. The set S need to be known so that it defines the domains of objects and the collection of operations Ω are defined on S.

Boolean Algebra of semi-algebraic objects gives us a concrete algebraic structure which is computable. Thus the theory of semi-algebraic sets provides a computable model for geometric modelling.

3.2.2 Semi-Algebraic Sets

The concept of Boolean algebra of sets is useful for geometric modelling since the basic idea of CSG is also to represent sets in R^n by Boolean combinations of primitive sets. If the geometric objects that we want to represent can be considered as sets then we would have Boolean algebra of geometric objects which allow us to define primitive geometric objects as sets and combining them together using Boolean operators.

In general, we would like to allow, as a CSG primitive, any set defined by a polynomial inequality or equality.

Let $Z[x_1, \ldots, x_n]$ denotes the set of polynomials in variables $x_1, \ldots, x_n$ with integral coefficients. A *semi-algebraic primitive* is a subset of R^n which admits some representation of the form

$$\{(x_1, \ldots, x_n) : p(x_1, \ldots, x_n) \; op \; 0\}$$

where $p(x_1, \ldots, x_n) \in Z[x_1, \ldots, x_n]$ and $op \in \{ \leq, <, =, \neq, >, \geq \}$. An expression of the form $p(x_1, \ldots, x_n) \; op \; 0$ is called an *atomic formula.*

For example, $\{(x, y, z) : x^2 + y^2 + z^2 - 4 \leq 0\}$ is a semi-algebraic primitive in R^3 with $x^2 + y^2 + z^2 - 4 \leq 0$ being the atomic formula.

A *semi-algebraic set* is a semi-algebraic primitive or a Boolean combination of semi-algebraic primitives. The semi-algebraic sets of R^n form a Boolean algebra with $+$ as set union ($\cup$), $\cdot$ as set intersection ($\cap$), $-$ as a set complement ($-$), 0 as an empty set (ϕ) and 1 as the universal set (R^n).

We can summarise the connection between Boolean logic and geometry as follow:

	Logic	Geometry
Primitive Objects	Conditions	Subsets of R^n
Operators	$\vee$	$\cup$
	$\wedge$	$\cap$
	$\neg$	$-$
	$\exists$	Projection

Boolean algebra of semi-algebraic sets provide a finite description of geometric objects and a set of operators capable of manipulating them. Semi-algebraic sets can be used as a CSG primitive. Object in CSG can be viewed as a set-theoretic composition of elementary semi-algebraic sets in R^n.

By definition, the Boolean combinations of semi-algebraic sets are closed under elementary set-theoretic operators; finite intersection, finite union and complement. We can use these semi-algebraic sets and its operators to define geometric objects.

For example, we can define a solid cylinder which has 2 units radius and 4 units height in xyz-space as:

$$A \cap B \cap C$$

where

$$\begin{aligned} A &= \{(x, y, z) : x^2 + y^2 - 4 < 0\} \\ B &= \{(x, y, z) : z > 0\} \\ C &= \{(x, y, z) : z < 4\} \end{aligned}$$

Not only that we can use semi-algebraic representation to represent solid objects, we can also use it to represent 2-dimensional object such as thin sheets or 1-dimensional object such as wires or objects with thin shell.

For example, we can define a thin-wall cylinder which has 2 units radius and 4 units height in xyz-space as:

$$A \cap B \cap C$$

where

$$\begin{aligned} A &= \{(x,y,z) : x^2 + y^2 - 4 = 0\} \\ B &= \{(x,y,z) : z > 0\} \\ C &= \{(x,y,z) : z < 4\} \end{aligned}$$

Extended Semi-Algebraic Sets

We can extend the CSG primitives beyond the semi-algebraic. It is often useful in applications to consider trigonometric and exponential functions.

Let $Z[x_1, \ldots, x_n, \sin(x_1), \ldots, \sin(x_n), \cos(x_1), \ldots, \cos(x_n), \exp x_1, \ldots, \exp x_n]$ denote the set of polynomials with integral coefficients in $x_1, \ldots, x_n$, and the sines, cosines, and exponentials of these variables. An *extended semi-algebraic primitive* is a subset of R^n which admits some representation of the form

$$\{(x_1, \ldots, x_n) : p(x_1, \ldots, x_n)\ op\ 0\}$$

where

$$p(x_1, \ldots, x_n) \in Z[x_1, \ldots, x_n, \sin(x_1), \ldots, \sin(x_n), \cos(x_1), \ldots, \cos(x_n), \exp x_1, \ldots, \exp x_n],$$

and $op \in \{ \leq, <, =, \neq, >, \geq \}$.

An *extended semi-algebraic set* is an extended semi-algebraic primitive or a Boolean combination of extended semi-algebraic primitives.

Boolean algebra of extended semi-algebraic sets, which includes trigonometric and exponential functions, has good expressive power. Not only that it is capable of representing static objects but it is also natural to describe motion constraints of objects in this form.

However, the subsets of R^n which can be represented by Boolean combinations of the extended primitives do not always have good closure properties. For example, although we can represent the trigonometric functions, we cannot, as far as we know represent the primitive $\{(x, y) : y - \sin(x^2) \leq 0\}$.

3.2.3 Closure Properties

Semi-algebraic sets have good closure properties. For example, they are closed under:

- Boolean operations
- Algebraic change of coordinate system
- Minkowski sum
- Projection

Additionally boundary of semi-algebraic sets are also semi-algebraic. Thus the semi-algebraic framework is very powerful and flexible. In practice, it is useful for a geometric model maker to have a large tool-box of transformations available.

On the other hand, the extended semi-algebraic sets as defined above are only seem to be closed under Boolean operations. Consequently, a CSG system with extended semi-algebraic primitives will have some structural limitations. This criticism applies to Svlis, described below, and almost any other system with non-algebraic primitives and only Boolean operations.

3.3 Geometric Modeller: Svlis

Svlis, developed by the Geometric Modelling Group at the University of Bath, is a CSG modeller using extended semi algebraic primitives, as described above.

The working group first created Svlis 3-dimensional CSG kernel modeller to be used as a tool to perform research into CSG modelling techniques. It was also aimed to provide a kernel capable of dealing with geometric object for higher level systems such as Computer Aided Design system.

To exploit the dimensional-independent property of CSG, Svlis was further developed so that it is capable of representing higher dimensional objects. The multi-dimensional Svlis is called Svlis-m but in this thesis we will refer to it as Svlis since Svlis-m is a superset of Svlis. Wise demonstrated in [68], the use of this multi-dimension CSG modeller to compute global C-space maps. This is only one application of Svlis, which is a general purpose system.

Svlis has a collection of built-in primitive shapes but also capable of representing geometric objects that can be expressed implicitly by polynomial inequalities. Object of zero thickness such as wires and sheets can be represented as well as solids. It also allows the use of *sine*, *cosine* and *exponential* functions when building descriptions of objects. More complicated objects are built by combining simple objects using operators of set-theory. Svlis provides four set-theory operators namely; union, intersection, difference, and symmetric difference [7].

To aid the efficiency of calculations concerning its objects, Svlis subdivides the region of interest into a collection of smaller sub-regions. The idea is based on the assumption that "representation of objects may be globally complicated but locally simple". The desirable outcome is that, each time the subdivision occurs, the smaller sub-region has a simpler object. Although it is likely, there is no guarantee that the object will be simpler in a smaller region of interest. It is also important that the union of all regions still represent the more complicated original model.

The method that Svlis employs is by using a combination of two processes. The first divides the original region of interest into many regions, each has the original object inside. This process is called *subdivision.* The second process reduces the number of primitives which made up the object in each region by systematically removing unnecessary primitives. This process is called *pruning* [69]. The subdivision process is applied recursively, each time with the pruning process to

simplify the object to its region, until a certain condition is met.

In Svlis, the subdivision process of the original regions of interest, or *boxes*, takes the form of Binary Spatial Division (BSD). That is the region is divided equally into two adjacent regions along an axis. Regions in Svlis are n-dimensional coordinate-aligned boxes defined by n closed intervals. These intervals are subsets of R defined in terms of end-points a and b. A closed interval $\{x | x \in R \text{ and } a \leq x \leq b\}$ usually denoted by $[a, b]$.

Many operators are defined for Svlis intervals, namely addition, subtraction, multiplication, division by real numbers, intersection, union and exponentiation to a positive integer power. The *sine*, *cosine* and *exp* are also defined. However, interval arithmetic is conservative so the resulting interval may be larger than it should be. Arithmetic on intervals is covered in great detail in [46].

The division decision in Svlis is to divide the longest side of the box. Since n-dimensional boxes are defined by n closed intervals, the division is done by dividing the longest interval into two equal parts. The original representation of the object is pruned to each of these smaller boxes and, hopefully, can be simplified. Each time the subdivision occurs, the box gets smaller and the simplification is "more likely" to happen.

Taking the simplification method of CSG objects from Svlis, the next section describes CSG system which allow the process of pruning and recursive subdivision on extended semi-algebraic sets with Boolean operator, using interval arithmetic.

3.4 CSG Systems

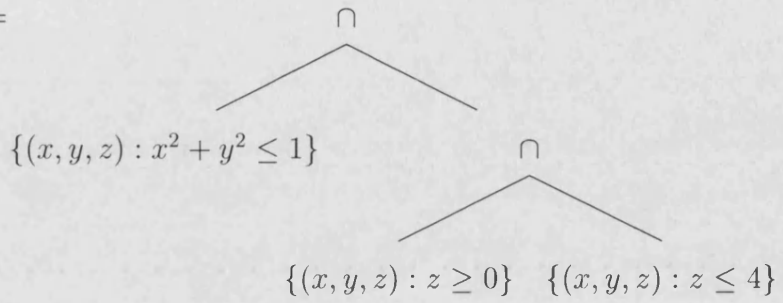
It is clear that, Boolean algebra of extended semi-algebraic sets provide a finite description of geometric objects and a set of operators capable of manipulating them. Call this system *CSG system.*

Complicated geometric objects in CSG system are built using Boolean combina-

tions of simple extended semi-algebraic primitives and can be represented by a data structure: a tree with Boolean operators on the internal nodes and extended semi-algebraic primitives at the leaves.

For example:

$Cylinder =$



To answer a query about an object represented in this way can be computationally expensive since every node and leaf has to be consulted. One way to overcome this problem is to use the pruning and recursive subdivision technique using interval arithmetic.

Interval Arithmetic

A finite interval on the real line is a subset of R defined in terms of end-points a and b. We are using closed interval $\{x | x \in R \ \wedge \ a \leq x \leq b\}$ denoted by $[a, b]$ which means that both end points belong to the subset.

An interval can be regarded as a finite region of one dimension. We refer to a region defined in this way as a box. For example, two intervals, one along each Cartesian coordinate axes represent 2-dimensional coordinate aligned box, which is an area between 4 line segments. In the same fashion, three interval can represent a cuboid.

The division of the space can be done by dividing the intervals which represent the box and interval arithmetic [46] can be used to determine how a primitive relates to the box. By substituting the corresponding variables of the atomic formulae which form the primitives by intervals, the output interval can be evaluated

according to the operator of the atomic formula. This process is called evaluating the primitive over a box which will be discussed later in Subsection 3.4.2.

3.4.1 Models and Boxes

In order to use the pruning and recursive subdivision technique, we introduce the concept of models and boxes.

Definition 3 *A CSG model M is a tree structure with Boolean operations on the internal nodes and atomic formulae on the leaves.*

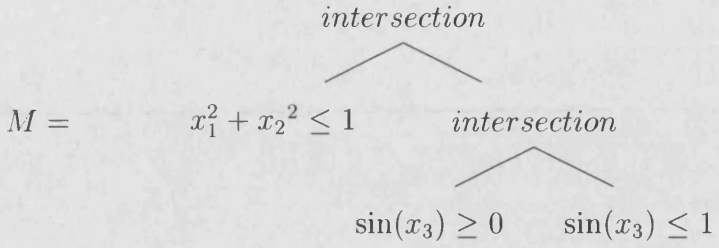
Additionally, we will assume some list $(x_1, \ldots, x_n)$ of variables which may appear in M. CSG object trees define subset of R^n but a CSG model trees such as M describe conditions of variable $(x_1, \ldots, x_n)$ available to them.

To distinguish between the two types of tree structure we introduce the following notations:

	CSG Tree	Model Tree
Primitives	Semi-algebraic Sets	Atomic Formulae
Operators	$\cup$	*union*
	$\cap$	*intersection*
	$-$	*complement*

The atomic formulae may define semi-algebraic primitives, or extended semi algebraic primitives. We might also wish to restrict the primitives to a subset of the semi-algebraic, such as, for example, linear half-spaces and cones. The geometric operations are n-ary union and intersection and unary complement.

For example, for a variable list (x_1, x_2, x_3)

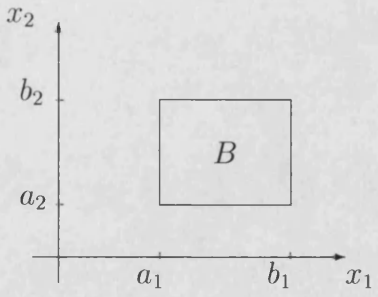


Definition 4 *Let B be a list of m closed intervals $([a_1, b_1], \ldots, [a_m, b_m])$. Call B a box. Each interval in B represents a coordinate-aligned edge of the box. The correspondence is determined by the ordering of the intervals and the ordering of variables in a specified ordered list.*

Suppose a box is m-dimensional and the ordered list of variables are $x_1, \ldots, x_n$ where $m \leq n$. By convention, the interval $[a_1, b_1]$ corresponds to the variable x_1, the interval $[a_2, b_2]$ corresponds to x_2, and the correspondence carry on respectively to $[a_m, b_m]$ which corresponds to x_m. The variables $x_{m+1}, \ldots, x_n$ are ignored.

Additionally, the ordered list of variables also label the coordinate axes of this set in R^n and if the length of the list of interval which defines the box is m, the box is said to be m-dimensional.

For example, suppose the ordered list of variables is (x_1, x_2, x_3, x_4) and the box B is $([a_1, b_1], [a_2, b_2])$. B is 2-dimensional coordinate-aligned box:



The box can also be represented in semi-algebraic form. For example, suppose the ordered list of variables is $x_1, \ldots, x_n$ and a box B of closed intervals is

$([a_1, b_1], \ldots, [a_m, b_m])$ where $m \leq n$. The box B can be written in semi-algebraic form as:

$$(x_1 \geq a_1 \wedge x_1 \leq b_1) \wedge \ldots \wedge (x_m \geq a_m \wedge x_m \leq b_m).$$

Definition 5 *Let M be a model with variable from $x_1, \ldots, x_n$ and let B be an m-dimensional box where $m \leq n$. Define the model M over the box B to be the set of points in B that satisfy M. Denote this set in R^n by (M, B).*

Since the set (M, B) is the collection of points in B that satisfy M, $(true, B)$ is the box B itself and $(false, B)$ is the empty set.

We will use a pair (M, B), where M is a model and B is a box, to define a subset of R^n. This (M, B) is our primitive CSG object; the building block of our geometric language which can be regarded as an extended semi-algebraic set.

For example, for a variable list (x_1, x_2, x_3) :

1. $M = x_1 < 0$, $B = ([-1, 1])$

$$(M, B) = \{x_1 : x_1 \geq -1 \wedge x_1 < 0\}.$$

2. $M = x_1 < 0$, $B = ([-1, 1], [-1, 1])$

$$(M, B) = \{(x_1, x_2)\} : (x_1 \geq -1 \wedge x_1 < 0) \wedge x_2 \geq -1 \wedge x_2 \leq 1\}$$

3. $M = complement(x_1 < 0)$, $B = ([-1, 1])$ defines

$$\{x_1 : x_1 \geq 0 \wedge x_1 \leq 1\}$$

4. $M = complement(x < 0)$, $B = ([-1, 1], [-1, 1])$

$$(M, B) = \{(x_1, x_2) : x_1 \geq 0 \wedge x_1 \leq 1 \wedge x_2 \geq -1 \wedge x_2 \leq 1\}$$

5.

$$M = \begin{array}{c} \textit{intersection} \\ \diagup \quad \diagdown \\ x_1{}^2 + x_2{}^2 - 1 \leq 0 \qquad \begin{array}{c} \textit{intersection} \\ \diagup \quad \diagdown \\ x_3 \geq -6 \qquad x_3 \leq 6 \end{array} \end{array}$$

$B = ([0,3],[0,4],[0,5])$

$$\begin{aligned} (M,B) &= \{(x_1,x_2,x_3) : x_1^2 + x_2^2 - 4 \leq 0 \wedge x_3 \geq -6 \wedge x_3 \leq 6 \wedge \\ &\quad x_1 \geq 0 \wedge x_1 \leq 3 \wedge x_2 \geq 0 \wedge x_2 \leq 4 \wedge x_3 \geq 0 \wedge x_3 \leq 5\} \\ &= \{(x_1,x_2,x_3) : x^2 + x_2^2 - 4 \leq 0 \wedge x_1 \geq 0 \wedge x_1 \leq 3 \wedge \\ &\quad x_2 \geq 0 \wedge x_2 \leq 4 \wedge x_3 \geq 0 \wedge x_3 \leq 5\}. \end{aligned}$$

It can happen as in example 2 and 4 above that (M,B) defines a set which does not depend on one or more of the variables from the variable list. Example 5 shows that this is necessary if we want intersection to be defined in a natural way. We have described a model as a tree. If we climb up the tree, the number of variables visible below us may change. We do not in general wish to change space every time this happens. Therefore, we work always in subsets of the box B. This will be discussed again later, when we consider projection.

The sets defined by (M,B) form a Boolean algebra of the subsets of B. This set definition is recursive on the structure of M. That is, if M is an atomic formula, (M,B) will be the subset of B in which M is *true* and it follows that:

- $(\textit{union}(M_1,M_2),B) = (M_1,B) \cup (M_2,B)$
- $(\textit{intersection}(M_1,M_2),B) = (M_1,B) \cap (M_2,B)$
- $(\textit{complement}(M),B) = B - (M,B)$

	$[-,0]$	$[0,+]$	$[-,+]$	$[-,-]$	$[+,+]$
$p=0$	*undecided*	*undecided*	*undecided*	$false$	$false$
$p<0$	*undecided*	$false$	*undecided*	*true*	$false$
$p\leq 0$	*true*	*undecided*	*undecided*	*true*	$false$

Table 3.1: Value of atomic formula according to the range of intervals

3.4.2 Evaluation

It is clear that, the box limit the scope in which the model is defined. Additionally, it also provides intervals which correspond to variables in the model from which we can determine the value of each atomic formula in the model tree by using interval arithmetic.

In order to reduce the number of atomic formulae of a model in a box, all the formulae that make up the model need to be evaluated. This is to determine the value of each atomic formula over the box and decide if it can be simplified. The evaluation is done by using interval arithmetic on each atomic formula. By substituting each variable of the formulae with the corresponding interval, the range of values of the function in the atomic formula can be calculated. This range will also be an interval. The value of the atomic formula can be evaluated according to the operator of the primitive. This process is referred to as *evaluating the formula over a box.* The value of the atomic formula after the substitution is either *true*, $false$ or *undecided.*

Table 3.1 shows the value of output intervals corresponding to the operator of the formula. The simplifications of models occur when some formula evaluate to *true* or $false$ and there is no simplification when the formula evaluated to *undecided.*

Since this set definition is recursive on the structure of M, when M is an atomic formula, (M, B) will be the subset of B in which M is *true.* That is, when the formula evaluated to *true* it can be interpreted as M is *true* inside the box B. That is, the intersection of the model and the box is the box itself. In this case

we replace the formula with *true*. When the formula evaluated to $false$, M is $false$ inside the box B and we can replace it with $false$. The original primitive is returned if the evaluation result is *undecided*.

For example, consider $M = 2x-3y+4z+12 \leq 0$ and the box $([-1,4],[1,3],[0,4])$. Over the box the M, in this case – an atomic formula, is:

$$\begin{aligned} 2([-1,4]) - 3([1,3]) + 4([0,4]) + 12 &\leq 0 \\ [-2,8] - [-9,-3] + [0,16] + [12,12] &\leq 0 \\ [1,33] &\leq 0 \end{aligned}$$

which evaluated to $false$.

This process can be thought of as a function which take an atomic formula P and a box B as the arguments and return either a value *true*, $false$ or the original primitive along with the box:

$$Eval(P,B) \rightarrow \{(true,B),(false,B),(P,B)\}.$$

An atomic formula of a model M will be evaluated to *undecided* if the atomic formula has an instance of a variable which does not correspond to an interval.

3.4.3 Pruning

The evaluation process may replace some unnecessary formulae of the model in the box by *true* or $false$ which can lead to a reduction of the number of formulae that made up a model. This is because a model consisted of unions, intersections and complements of formulae. Once each formula in the model has been evaluated and some replaced with *true* or $false$, we can work up the model tree applying the operators to each leaf and achieve some simplification.

The pruning process ensures that inside the box there are no unnecessary formulae while the simplified model is still representing the original one. We can simplify

the model over the box by applying the rules of the operators to the primitives.

The simplification rules of the CSG Boolean operators are:

- $union(undecided, true) = true$
- $union(undecided, false) = undecided$
- $intersection(undecided, true) = undecided$
- $intersection(undecided, false) = false$
- $complement(true) = false$
- $complement(false) = true$
- $complement(undecided) = undecided$

The process of pruning the model to the box takes a model and a box (M, B) and produces another model over the same box (M', B). In the box B, M' is either the same as M or simpler than M. Also, (M, B) and (M', B) define the same set.

The pruning procedure can be expressed in terms of a recursive algorithm (See Algorithm 1). It starts at the root of the model and works down to all the leaves. While the node is still an operator, the evaluation is deferred by calling the procedure $Prune()$ again.

Since CSG operators are n-ary union, n-ary intersection and unary complement, the algorithm can exploit many known properties. For example, it can take into account the simplification that can be made in the case where one of the operand of *union* simplified to *true* and similarly, where one of the operand of *intersection* simplified to $false$. If the node is not one of CSG operators then we have reached the leaf and procedure $Eval()$ is called to evaluate that particular formula to the box.

Algorithm 1 $Prune(M, B)$

Input: (M, B)
Output: (M', B)
Ensure: (M', B) defines the same set as (M, B)

```
S ← φ
if (M = union{M_1, ..., M_k}) then
  for i = 1 to k do
    (M_i', B) ← Prune(M_i, B)
    if (M_i' = true) then
      return(true, B)
    else if (M_i' ≠ false) then
      S ← union(S, M_i')
  if (S = φ) then
    return(false, B)
  else
    return(S, B)
else if (M = intersection{M_1, ..., M_k}) then
  for i = 1 to k do
    (M_i', B) ← Prune(M_i, B)
    if (M_i' = false) then
      return(false, B)
    else if (M_i' ≠ true) then
      S ← intersection(S, M_i')
  if (S = φ) then
    return(true, B)
  else
    return(S, B)
else if (M = complement{M_1}) then
  (M_1', B) ← Prune(M_1, B)
  if (M_1' = true) then
    return(false, B)
  else if (M_1' = false) then
    return(true, B)
  else
    return(complement{M_1'}, B))
else
  return(Eval(M, B), B)
```

For example, let consider $M = union(M_1, intersection(M_2, M_3))$ where:

$$\begin{aligned} M_1 &= intersection(x - y \geq 0, x \leq 3, y \geq 1) \\ M_2 &= (x-5)^2 + (y-6)^2 - 1 \leq 0 \\ M_3 &= intersection(x \geq 4, x \leq 6, y \geq 4) \end{aligned}$$

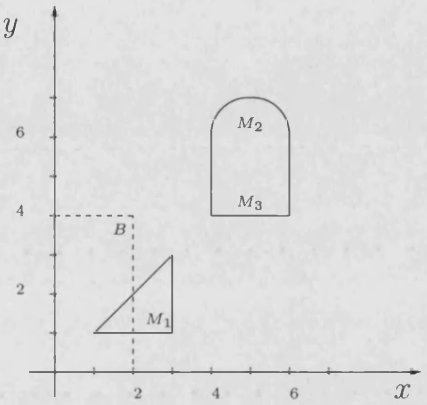


Figure 3-1: M_1, M_2, M_3 relative to the box B.

and the variable list is (x, y) (Figure 3-1). Prune M to the box $([0,2],[0,4])$ by evaluating each atomic formula:

$M_1 = intersection(x - y \geq 0, x \leq 3, y \geq 1)$ where $x = [0,2]$ and $y = [0,4]$.

$$\begin{aligned} M_1 &\approx intersection([0,2] - [0,4] \geq 0, [0,2] \leq 3, [0,4] \geq 1 \\ &\approx intersection(undecided, true, undecided) \\ &\approx intersection(undecided, undecided) \end{aligned}$$

Hence $M_1 = intersection(x - y \geq 0, y \geq 1)$

$M_2 = (x-5)^2 + (y-6)^2 - 1 \leq 0$

$$\begin{aligned} M_2 &\approx ([0,2] - 5)^2 + ([0,4] - 6)^2 - 1 \leq 0 \\ &\approx [9,25] + [4,36] - [1,1] \leq 0 \\ &\approx false \end{aligned}$$

Hence $M_2 = false$

$M_3 = intersection(x \geq 4, x \leq 6, y \geq 4)$

$$\begin{aligned} M_3 &\approx intersection([0,2] \geq 4, [0,2] \leq 6, [0,4] \geq 4) \\ &\approx intersection(false, true, undecided) \\ &\approx false \end{aligned}$$

Hence $M_3 = false$

Hence we have, $M = intersection(x - y \geq 0, y \geq 1)$.

3.4.4 Recursive Subdivision

Since we are representing boxes using intervals, we consider using coordinate-aligned box division since it is straightforward to perform subdivision technique and to determine adjacencies between boxes.

We consider two simple subdivision techniques, namely, grid-divisions and recursive subdivision. For the grid division technique, a box is divided into two or more sub-boxes of a specified size. This size is usually referred to as grid resolution. For example, let B be a box define by $([-1,7],[3,5],[0,2])$. For a grid resolution of 2 unit, B can be divided into:

$$B_1 = ([-1,1],[3,5],[0,2]) \quad B_2 = ([1,3],[3,5],[0,2])$$
$$B_3 = ([3,5],[3,5],[0,2]) \quad B_4 = ([5,7],[3,5],[0,2])$$

In contrast, recursive subdivision usually divide a box into two sub-boxes and the divisions carry on recursively until the termination conditions are met. There are two subdivision decision strategies; *blind* and *adaptive*. The blind strategy determines the position of the subdivision by relatively fixing the point for each sub-division. For example, subdividing at a mid-point between two points. The adaptive strategy determine subdivision points by taking into account the contents of the box.

Regardless of the division method and strategy, when a box is divided, we get two or more adjacent sub-boxes with the same model inside. Each model can then be simplified over its own box by the pruning process and hopefully will be simpler than the original model. In this way, at any point, the union of all sub-boxes with their models results in the original box with the original model.

Since the set definition of (M, B) is recursive on the structure of M, if $B = union(B_1, B_2)$ it follows that:

$$(M, B) = union((M, B_1), (M, B_2))$$

For example, given a box B and a model $M = union(intersection(M_1, M_2), M_3)$, if the box is subdivided into two sub-boxes, B_1 and B_2 along one of its coordinate, then:

$$\begin{aligned}(M, B) &= (union(intersection(M_1, M_2), M_3), B) \\ &= union(\\ &\quad union(intersection((M_1, B_1), (M_2, B_1)), (M_3, B_1)), \\ &\quad union(intersection((M_1, B_2), (M_2, B_2)), (M_3, B_2)))\end{aligned}$$

The algorithm, which divides the longest side of the box, can be expressed in algorithmic form as in Algorithm 2.

Algorithm 2 $SubDivide(M, B)$

Input: M, B

Output: $(M, B_1), (M, B_2)$

$position \leftarrow MaxSidePosition(B)$
$interval \leftarrow Part(B, position)$
$lower \leftarrow LowerEnd(interval)$
$upper \leftarrow UpperEnd(interval)$
$midpoint \leftarrow lower + Size(interval)/2$
$B_1 \leftarrow Part(B, position) \leftarrow [lower, midpoint]$
$B_2 \leftarrow Part(B, position) \leftarrow [midpoint, upper]$
$return\ (M, B_1), (M, B_2)$

After the pruning process, the union of the sets defined by (M, B_1) and (M, B_2) is the same as the set defined by (M, B). The division can be carried on recursively along with the pruning process until some conditions are met. There are a few options to consider as a termination condition of the process. For example, as in Svlis, the recursive subdivision could stop when all the boxes are sufficiently small. It is also possible to determine if the model is simple enough that is there are a certain number of atomic formulae left in the model.

The Algorithm 3 recursively subdivides the longest side of the box until the box is small enough or the model is simple enough.

Algorithm 3 *RecurSubDivision*(M, B)

Input: M, B

Output: $(M_1, B_1), \ldots, (M_k, B_k)$

$(M', B) \leftarrow Prune(M, B)$
if $IsSimple(M')$ *or* $IsSmall(B)$ **then**
$return(M', B)$
else
$(M, B_1), (M, B_2) \leftarrow SubDivide(M, B)$
$return(RecurSubDivision(M, B_1), RecurSubDivision(M, B_1))$

When we extend the primitive to extended semi-algebraic sets, pruning and recursive subdivision are still applicable. This is because the evaluation is done by interval arithmetic which *sine*, *cosine* and *exp* functions are also defined.

3.5 CSG Approach to Spatial Planning

A major approach of CSG to spatial planning is to construct C-space obstacles, which can be regarded as geometric objects in C-space, as a Boolean combination of CSG primitives. Each point inside this object represents the position of the actual object that causes collision with the obstacles in the Workspace. Given C-space obstacles, Find-space and Find-path correspond to the simpler problems of finding a single point - a position of the object, and a path - a sequence of

position of the object, outside the obstacles.

The property of C-space obstacle in which:

$$CO_P(O_1 \cup O_2) \equiv CO_P(O_1) \cup CO_P(O_2)$$

where P is the object and O_i are obstacles, gives rise to the idea of generating C-space obstacle of many simple objects and combining the results together. The validity of this property holds regardless of convexity or connectedness of the obstacles [8]. This property is particularly useful with the CSG representation of the C-space obstacles since it is natural to build complicated object using Boolean combinations of many simple ones.

Wise investigated in [68] the application of CSG to the problem of generating the global C-space map and demonstrated two approaches which semi-algebraic CSG can be used to compute global map of the C-space for a system of rigid bodies. The first is an approximate calculation of C-space obstacles and the second is a precise representation of C-space obstacles by formulating the contact surfaces analytically. The potential for combining the two approaches are also highlighted.

Despite the dimensionally independent nature of CSG models, Wise and Bowyer [7] is the only map-maker to exploit the representation.

3.6 Summary

We gave a brief description of Constructive Solid Geometry and summarised the mathematical framework of the semi-algebraic approach to CSG. The idea of CSG was also introduced along with the key technique used by Svlis CSG modeller. We gave an overview of semi-algebraic CSG approaches to spatial planning by using CSG technique to represent C-space obstacles. The algorithm for pruning and recursive subdivision the extended semi-algebraic models were also described.

Chapter 4

Quantifier Elimination

In this chapter we give an overview of the quantifier elimination problems and give a brief report on the complexity estimates of the quantifier elimination process. We also describe briefly the idea of Cylindrical Algebraic decomposition which can be applied to quantifier elimination. Finally, we introduce the quantifier elimination approach to spatial planning.

4.1 Quantifier Elimination Problems

Let L_R be the first order language of the ordered field of the reals. A *formula* of this language is the expression which are built up from atomic formulae using the logical operators $\wedge, \vee, \neg$. An *atomic formula* is the expression of the form $p(x_1, \ldots, x_n)$ *op* 0 where $p(x_1, \ldots, x_n)$ is a polynomial in variables $x_1, \ldots, x_n$ with integral coefficients and $op \in \{>, \geq, \neq, <, \leq\}$. Additionally, some or all of the variables in a formula may be quantified over the field by universal $(\forall x)$ and existential $(\exists x)$ quantifier.

For example,

$$A = (\exists x_1)(\forall x_2)[((p_1(x_1, x_2, x_3) < 0) \vee (p_2(x_1, x_2, x_3) < 0)) \wedge (p_3(x_1, x_2, x_3) = 0)]$$

is a formula of the language L_R.

An occurrence of a variable x in A is *bound* if it occurs in a sub-formula B of the form $(\exists x)B$ or $(\forall x)B$. An occurrence of a variable is *free* if it is not bound. In the above example, occurrences of variable x_1 and x_2 are bound and the occurrence of the variable x_3 is free. Suppose the formula $A(x_1, \ldots, x_n)$ has free variables among $x_1, \ldots, x_n$. This formula is semi-algebraic and defines a subset of R^n: $\{(x_1, \ldots, x_n) : A(x_1, \ldots, x_n)\}$.

A formula in which all variables are quantified is called a *sentence* which has a definite truth value. When free variables are substituted by specific values leaving only bound variable in the formula, it becomes a sentence. A set of values is a *solution* for the formula if the sentence, obtained by substituting all variables in the formula by the values, is true. Two formulae are *equivalent* if they have the same solutions.

In 1930, Tarski showed in [64] that all quantified formulae can also be defined without quantifiers and presented an algorithmic quantifier elimination (QE) method. The algorithm accepts any formula of real closed field as an input and outputs an equivalent formula containing the same variables with no quantifiers. The output formula is true for the same values of its free variables as the input formula. If the input formula contains free variables then the output formula expresses a necessary and sufficient algebraic condition of the input formula to hold.

For example, applying the quantifier elimination to the formula of L_R:

$$a \neq 0 \wedge (\exists x)[ax^2 + bx + c = 0]$$

produces the well-known necessary and sufficient condition

$$a \neq 0 \wedge b^2 - 4ac \geq 0$$

where a, b, c are free variables.

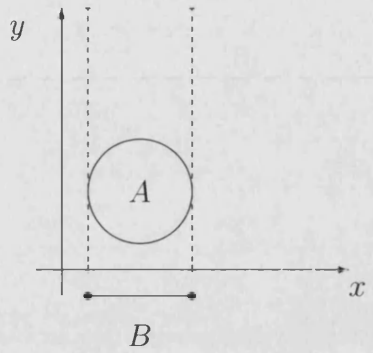


Figure 4-1: Existential quantifier corresponds to a geometric projection.

4.1.1 Existential Quantifiers as Geometric Projections

We use semi-algebraic sets which are subsets of some R^n defined by a finite number of polynomial equations and inequalities to represent n-dimensional object in n-dimensional space.

We can regard a set defined by a formula $B = (\exists y)A$ as a set of points at which every atomic formula in A is true. In this way, existential quantifiers correspond to geometric projections. The set of points defined by B is a projection of a set defined by A.

For example, let A define an area inside the circle

$$A = \{(x, y) : (x - 3)^2 + (y - 3)^2 - 4 \leq 0\}.$$

Suppose $B = (\exists y)A$. The set of points defined by B is a projection of a set defined by A. The projection is parallel to the y axis onto the space of other variables of A; in this case a 1-dimensional space of x (See Figure 4-1).

4.2 Complexity Estimates

Renegar presented in [56] a brief survey of some complexity highlights for quantifier elimination methods. We summarise the main result which concerns our application as follow:

The elimination of a block of n_0 existential quantifiers has complexity bounded approximately by

$$(md)^{n_0 n_1 C} Cost$$

where C is some constants, n_1 is the number of free variables and d is the maximum degree of m different polynomials involved. The Cost term is generally negligible compare to $(md)^{n_0 n_1 C}$

Later in our practical experiments we are considering the case where $n_0 = 2$. We do not have a good estimate for the exponent multiplier C. We expect that C is significantly larger than 1. We expect complexity for our problem to increase as some power of md.

Beyond linear case and a few variables, quantifier elimination is very difficult, or produces such huge output as to be hard to manage. Additionally, it is only possible in principle to eliminate quantifier if the bound variables only occur algebraically.

4.3 Application of Quantifier Elimination

Here are some examples.

- A unit square is defined by the formula:

$$x \geq 0 \wedge x \leq 1 \wedge y \geq 0 \wedge y \leq 1$$

An ellipse is defined by the formula:

$$x^2/4 + y^2/9 - 1 \leq 0$$

We can define the Minkowski sum of the unit square and the ellipse as:
$(\exists\ x, y, z, w)$
$[u = x+z \wedge v = y+w \wedge x \geq 0 \wedge x \leq 1 \wedge y \geq 0 \wedge y \leq 1 \wedge z^2/4+w^2/9-1 \leq 0]$
According to the Tarski Theorem, this set can also be defined without quantifiers.

- Let S be a subset of R^n which is defined by a formula $A(x_1, \ldots, x_n)$. Define *closure*(S) to be the set of points in R^n which have points of S arbitrarily near. We can define this set *closure*(S) by a formula of L_R as:

$$\begin{aligned} \mathit{closure}(S) \ = \ & \{(y_1, \ldots, y_n) : (\forall \epsilon > 0)(\exists x_1, \ldots, x_n)[A(x_1, \ldots, x_n) \wedge \\ & (x_1 - y_1)^2 + \cdots + (x_n - y_n)^2 < \epsilon]\} \end{aligned}$$

  Therefore, according to the Tarski theorem, if S is semi-algebraic, so is *closure*(S)

- Define *boundary*(S) to be *closure*$(S) \cap$ *closure*(*complement*(S)). As before, Tarski's theorem implies that if S is semi-algebraic, so is the boundary of S.

4.4 Cylindrical Algebraic Decomposition

In 1973, Collins [13] has used Cylindrical Algebraic Decomposition (CAD) to eliminate quantifiers. The CAD method for quantifier elimination consists of three main phases.

The first phase extracts the polynomials occurring in the input formula and factoring them into irreducible factors, assuming that each atomic formula of the input formula is of the form $p = 0$ or $p < 0$ where p is a multivariate polynomial with integer coefficients.

The second phase constructs a decomposition of real r-dimensional space, where r is the number of variables in the formula, into a finite number of connected regions, called *cells*. Each polynomial in all cells is invariant in sign. These cells are arranged in a certain cylindrical manner. From this cylindrical decomposition, it is then quite straightforward to apply the quantifiers by using a sample point from each cell to determine the invariant truth value of the input formula in that cell. This application of quantifiers reveals which cells in the subspace of the free variables are true.

The final phase constructs an equivalent quantifier-free formula from this knowledge. In Collins' original method, this problem was solved by a method called augmented projection that provided a quantifier-free formula for each of the true cells.

In 1990, Hong [29] has devised a generally more efficient method but only appears to work in most cases. In 1995, Hong [30] also proposed and implemented a more efficient approximate quantifier elimination using interval arithmetic. The algorithm was later improved and implemented by Ratschan [51] [52] [53].

4.5 Quantifier Elimination Approach to Spatial Planning

The mathematical and computational structures of the spatial planning problem when stated in algebraic terms are reasonably well-understood [11] [58]. Objects defined by semi-algebraic sets are highly flexible and expressive but they are computationally expensive to compute [11] [16].

Existential quantifiers, which correspond to geometric projection, and extended semi-algebraic sets with Boolean operators can be used to describe C-space obstacles and Freespace. Several methods based on generating a cylindrical cell decomposition of Freespace were first proposed by Schwartz and Sharir [58].

We can outline the method as follow:

1. Represent C-space obstacle in quantified form.
2. Decompose C-space into semi-algebraic cells so that each cell is either entirely contained in the C-space obstacle or disjoint from it.
3. Determine adjacency relation between cells in C-space obstacle. Represent obstacle connectivity as finite graph.
4. Solve Findpath problem by standard graph searching techniques.

Both step 2 and 3 are difficult in practice. For example, one approach to step 3 is to decide if cell C_1 is adjacent to cell C_2. Cell C_1 and C_2 are adjacent if $closure(C_1)$ intersects C_2 or if C_1 intersects $closure(C_2)$. The closures can be expressed in quantified form, and then the quantifiers can be eliminated.

The set of collision-free points is a semi-algebraic set that can be determined by QE. It is then necessary to decide all cells adjacencies in the CAD in order to determine whether the two points are in the same component.

Attempts to solve simple two dimensional spatial planning problem using CAD to eliminate quantifiers, for example, by Davenport [14], Davenport et al. [15], Kalkbrener and Stifter [36], McCallum [45], Sturm and Weispfenning [63], were unsuccessful because of excessive computational resource requirement.

4.6 Summary

We gave an overview of the quantifier elimination methods and the cylindrical algebraic decomposition. We gave a brief report on the complexity estimates of the process of quantifier elimination and introduced the quantifier elimination approach to spatial planning.

Chapter 5

Using Quantifier Elimination

In this chapter we present methods of constructing C-space obstacle using existential quantifiers and boundary formation. We introduce the applications of pruning and spatial subdivision techniques to speed-up the computation of quantifier-free representation. We also present some experimental results and discussions.

5.1 Representing C-space Obstacles

Semi-algebraic sets are closed under finite union, intersection, and negation. Additionally, boundary and projection of semi-algebraic sets are also semi-algebraic. Thus C-space obstacles can be built not only from Boolean combinations of semi-algebraic primitives but also with quantifier and boundary operator.

5.1.1 Using Existential Quantifiers

The idea is to characterise the position and the geometric constraints of the object due to the obstacles by some conditions imposed on the entire object by the presence of the obstacle in the same Workspace. This can be done by stating

the conditions of every point of the object using quantifiers.

An Example

Consider the case of an object P and obstacles O in R^2. Assuming that P and O are in the same coordinate frame and the reference point is inside the object. For a point $z, w \in R^2$, define P and O as subsets of the Workspace R^2 where:

$P = \{(z, w) : p(z, w)\}$ where $p(z, w)$ is some conditions for (z, w) and

$O = \{(z, w) : o(z, w)\}$ where $o(z, w)$ is some conditions for (z, w).

From the definition of C-space obstacle it follows that:

> There exist a point in the object such that, if this point translated and rotated about its origin, then it will be in the obstacle.

The collection of such points form C-space obstacles. C-space obstacles can be defined using a combination of existential quantifiers and usual Boolean operators on semi-algebraic primitives. If P is allowed to translate in each dimension by x and y respectively, and rotate freely around its reference point by θ then C-space obstacles of P due to O can be defined as a combination of a translation and a rotation as:

$$CO_P(O) = \{(z, w, \theta) \ : \ \exists z \exists w [p(z, w) \ \wedge o(x + (z\cos\theta - w\sin\theta), y + (z\sin\theta + w\cos\theta))]\}. \tag{5.1}$$

Consequently, we can obtain the equivalent quantifier-free semi-algebraic formula of the C-space obstacles by eliminating the quantifiers. In the above example, the quantifier-free result of $CO_P(O)$ defines a geometric object; the C-space obstacle, defined as a collection of points $(x, y, \theta) \in R^3$.

Representing C-space obstacles in this way results in more variables than the degrees of freedom of the problem. The Boolean combination of the formulae

with variables z, w, x, y, θ that built up $CO_P(O)$ define a geometric object which we refer to as the *Omnimodel*[†] and the space as the *Omnispace.*

General Case in R^n

Let P represent an object and O represent the obstacle in R^m. Suppose the variables available to P and O are $(z_1, \ldots, z_m)$. Allow the object to translate freely in all directions by $(t_1, \ldots, t_m)$. Also allow the object to rotate freely about its reference point around all axis by $(\theta_1, \ldots, \theta_n)$ where $n = \frac{1}{2}m(m-1)$). We can represent the C-space obstacles $CO_P(O)$ in quantified form as:

$$\{(t_1, \ldots, t_m, \theta_1, \ldots, \theta_n) \ : \ \exists(z_1, z_2)[P(z_1, z_2) \wedge O \begin{pmatrix} t_1 + (z_1 \cos\theta_1 - z_2 \sin\theta_1), \\ t_2 + (z_1 \sin\theta_1 + z_2 \cos\theta_1) \end{pmatrix}]\}$$

where $m = 2$ and $n = 1$, or

$$\{(t_1, \ldots, t_m, \theta_1, \ldots, \theta_n) \ : \ \exists(z_1, z_2, z_3)[P(z_1, z_2, z_3) \wedge O \begin{pmatrix} t_1 + (z_1 \cos\theta_1 - z_2 \sin\theta_1) + (z_3 \sin\theta_3 + z_1 \cos\theta_3), \\ t_2 + (z_2 \cos\theta_2 - z_3 \sin\theta_2) + (z_1 \sin\theta_1 + z_2 \cos\theta_1), \\ t_3 + (z_3 \cos\theta_3 - z_1 \sin\theta_3) + (z_2 \sin\theta_2 + z_3 \cos\theta_2) \end{pmatrix}]\}$$ where $m = 3$ and $n = 3$, or

$$\{(t_1, \ldots, t_m, \theta_1, \ldots, \theta_n) \ : \ \exists(z_1, z_2, z_3, z_4)[P(z_1, z_2, z_3, z_4) \wedge O \begin{pmatrix} t_1 + (z_1 \cos\theta_1 - z_2 \sin\theta_1) + (z_1 \cos\theta_2 - z_4 \sin\theta_2) + (z_3 \sin\theta_4 + z_1 \cos\theta_4), \\ t_2 + (z_4 \sin\theta_6 + z_2 \cos\theta_6) + (z_1 \sin\theta_1 + z_2 \cos\theta_1) + (z_2 \cos\theta_3 - z_3 \sin\theta_3), \\ t_3 + (z_3 \cos\theta_4 - z_1 \sin\theta_4) + (z_3 \cos\theta_5 - z_4 \sin\theta_5) + (z_2 \sin\theta_3 + z_3 \cos\theta_3), \\ t_4 + (z_3 \sin\theta_5 + z_4 \cos\theta_5) + (z_4 \cos\theta_6 - z_2 \sin\theta_6) + (z_1 \sin\theta_2 + z_4 \cos\theta_2) \end{pmatrix}]\}$$

where $m = 4$ and $n = 6$.

Finally in the general case where there are m degrees of freedom for the translations and $n = \frac{1}{2}m(m-1)$ degrees of freedom for the rotations, we can represent the C-space obstacles of the object due to the presence of the obstacle in quan-

[†]Originated by Woodwark et al. [48] [49] [50]

tified form as:

$$\{(t_1,\ldots,t_m,\theta_1,\ldots,\theta_n) \ : \ \exists(z_1,\ldots,z_m)[P(z_1,\ldots,z_m)\wedge \\ O\begin{pmatrix} t_1+(z_1\cos\theta_1-z_2\sin\theta_1)+\cdots+(z_m\sin\theta_n+z_1\cos\theta_n), \\ \ldots, \\ t_m+(z_m\cos\theta_n-z_1\sin\theta_n)+\cdots+(z_{m-1}\sin\theta_{n-1}+z_m\cos\theta_{n-1}) \end{pmatrix}]\} \tag{5.2}$$

By using quantifier elimination on the representations 5.2 in the Omnispace R^{m+n}, we obtain the quantifier-free extended semi-algebraic representation of the C-space obstacle in $(t_1,\ldots,t_m,\theta_1,\ldots,\theta_n)$.

Special Cases

We may also consider special cases which may arise. For example, when the object may not rotate, when the object may not translate in all possible dimensions, or when the object is only a point in the Workspace.

Consider the following examples.

Example 1 *Let P be the object and O be the obstacle, both in R^2. Assuming that P and O are in the same coordinate frame and for a point $(z,w) \in R^2$ let P and O be a subset of R^2 where:*

$$P = \{(z,w) : z^2+w^2 \leq 1\}$$

and

$$O = \{(z,w) : (z-5)^2+(w-5)^2 \leq 1\}.$$

Allow P to translate in z and w dimension by x and y respectively. C-space obstacle of P due to O can be defined as:

$$CO_P(O) = \{(x,y) : \exists z\exists w[z^2+w^2 \leq 1 \ \wedge (x+z+5)^2+(y+w+5)^2 \leq 1]\}$$

which can be regarded as a geometric object in R^2.

The equivalent quantifier-free semi-algebraic formula C-space obstacles after the eliminate the quantifiers is: $\{(x,y) \in R^2 : (x-5)^2+(y-5)^2 \leq 4\}$[†]. It is clear that when n-dimensional object may translate in all dimensions but may not rotate, the object has n degrees of freedom, thus the C-space is also n-dimensional. The C-space obstacle can be defined by:

$$CO_P(O) = \{(x_1,\ldots,x_n) : \exists z_1,\ldots,z_n[p(z_1,\ldots,z_n) \ \wedge \ o(z_1+x_1,\ldots,z_n+x_n)]\}$$

Example 2 *Let P be the object and O be the obstacle in R^2. Assuming that P and O are in the same coordinate frame and for a point $(z,w) \in R^2$, P and O are a subset of some R^2 where:*

$$P = \{(z,w) : z^2 + w^2 \leq 1\}$$

and

$$O = \{(z,w) : (z-5)^2 + w^2 \leq 1\}.$$

Allow P to translate in z dimension by x. C-space obstacle of P due to O can be defined as:

$$CO_P(O) = \{x : \exists z \exists w[z^2 + w^2 \leq 1 \ \wedge (x+z+5)^2 + (w+5)^2 \leq 1]\}$$

which can be regarded as a geometric object in R^1.

Example 3 *Let P be the object and O be the obstacle in R^2. Assuming that P and O are in the same coordinate frame. For a point $(z,w) \in R^2$, P and O are a subset of some R^2 where:*

$$P = \{(z,w) : z = 0 \wedge w = 0\}$$

and

$$O = \{(z,w) : z^2 + w^2 \leq 1\}.$$

Allow P to translate in z and w dimension by x and y respectively, and rotate

[†]Result from Partial Cylindrical Algebraic Decomposition Version 15 (Interactive) May, 1996 by Hoon Hong (hhong@risc.uni-linz.ac.at), Research Institute for Symbolic Computation.

freely around its reference point by θ. C-space obstacle of P due to O can be defined as:

$$\begin{aligned} CO_P(O) &= \{(x,y) : \exists z \exists w [z = 0 \wedge w = 0 \wedge \\ &\quad (x + (z\cos\theta - w\sin\theta))^2 + (y + (z\sin\theta + w\cos\theta))^2 \leq 1]\} \\ &= \{(x,y) : \exists z \exists w [(x+0)^2 + (y+0)^2 \leq 1]\} \\ &= \{(x,y) : x^2 + y^2 \leq 1\}. \end{aligned}$$

The equivalent quantifier-free semi-algebraic formula C-space obstacles after eliminating the quantifiers is: $\{(x,y,\theta) \in R^3 : x^2 + y^2 \leq 1\}$. Since a point object which may translate in n dimensions has two degrees of freedom, the C-space is n-dimensional. The C-space obstacle of this n-dimensional case can be defined by:

$$\begin{aligned} CO_P(O) &= \{(t_1,\ldots,t_m,\theta_1,\ldots,\theta_n) : \exists (z_1,\ldots,z_n)[o(t_1,\ldots,t_n)]\} \\ &= \{(t_1,\ldots,t_m,\theta_1,\ldots,\theta_n) : o(t_1,\ldots,t_n)\}. \end{aligned}$$

5.1.2 Incorporating Boundary Formation

A set is *closed* if it contains its boundary. We may exploit the boundary formation to form the C-space obstacles if we assume that the obstacles O defined by closed sets. By assuming that the object P is connected and the reference point is inside the object it follows that the object is always either entirely outside the obstacle or entirely inside the obstacle or is intersected with at least one edge of the obstacle. Thus the position of the object is impossible if and only if, either there is a point in the object such that, if this point translated and rotated around the reference point, it will be on the boundary of the obstacle, or the object is entirely in the obstacle. The collection of such point form the C-space obstacle.

The above statement implies that C-space obstacles consist of two disjoint sets, $CO_P(O) = S_1 \cup S_2$ where S_1 is the set of points which correspond to the configuration when the object intersect the boundary of the obstacle whereas S_2 is the

set of points which correspond to the configuration when the object is entirely inside the obstacle.

We can describe S_1 as:

> There exist a point in the object such that, if this point translate and rotate about its origin, then it will be in the boundary of the obstacle.

We can define S_1 using existential quantifiers and boundary as:

$$\{(t_1,\ldots,t_m,\theta_1,\ldots,\theta_n) \ : \exists(z_1,\ldots,z_m)[P(z_1,\ldots,z_m) \wedge Boundary(O\begin{pmatrix} t_1+(z_1\cos\theta_1 - z_2\sin\theta_1)+\cdots+(z_m\sin\theta_n+z_1\cos\theta_n), \\ \ldots, \\ t_m+(z_m\cos\theta_n - z_1\sin\theta_n)+\cdots+(z_{m-1}\sin\theta_{n-1}+z_m\cos\theta_{n-1}) \end{pmatrix})]\}. \tag{5.3}$$

In contrast, S_2 define a collection of points where the object is entirely in the obstacle which can be describe as:

> There does not exist a point in the object such that, if this point translate and rotate about its origin, then it will not be in the obstacle.

We can define S_2 using existential quantifiers as:

$$\{(t_1,\ldots,t_m,\theta_1,\ldots,\theta_n) \ : \ \neg\exists(z_1,\ldots,z_m)[P(z_1,\ldots,z_m)\wedge \neg O\begin{pmatrix} t_1+(z_1\cos\theta_1 - z_2\sin\theta_1)+\cdots+(z_m\sin\theta_n+z_1\cos\theta_n), \\ \ldots, \\ t_m+(z_m\cos\theta_n - z_1\sin\theta_n)+\cdots+(z_{m-1}\sin\theta_{n-1}+z_m\cos\theta_{n-1}) \end{pmatrix}]\}. \tag{5.4}$$

However, it is not necessary to compute $S_1 \cup S_2$ as two disjoint sets.

We can define another set, S_3, as the set of points which correspond to the configuration when parts of the object is intersect the obstacle. This set is equivalent to the case of point object as in Example 3, where the reference point of the object translate inside the obstacle.

S_3 can be represented as:

$$\begin{aligned} &= \{(t_1,\ldots,t_m,\theta_1,\ldots,\theta_n) \ : \ \exists(z_1,\ldots,z_m)[p(z_1=0,\ldots,z_m=0) \ \wedge \\ &\quad (o(t_1,\ldots,t_m,\theta_1,\ldots,\theta_n))]\} \\ &= \{(t_1,\ldots,t_m) : o(t_1,\ldots,t_m)\}. \end{aligned}$$

It is clear that $S_2 \subseteq S_3$ but $S_1 \cup S_2 = S_1 \cup S_3$ since it is true that if $(t_1,\ldots,t_m)$ is in $CO_P(O)$ and the translated reference point is not in the obstacle, then the object must intersect the boundary of the obstacle, since the object is connected. On the other hand, if the object intersects one of the boundary of the obstacle and the obstacle is closed or if the translated reference point is in the obstacle then $(t_1,\ldots,t_m)$ is in $CO_P(O)$.

Theorem 3 *If the object is connected and the obstacle is closed then we can define C-space obstacle of P due to O using quantifier and boundary operator as:*

$$\begin{aligned} &\exists(z_1,\ldots,z_m)[P(z_1,\ldots,z_m) \wedge Boundary(\\ &O\left(\begin{array}{c} t_1+(z_1\cos\theta_1 - z_2\sin\theta_1)+\cdots+(z_m\sin\theta_n+z_1\cos\theta_n),\ldots, \\ t_m+(z_m\cos\theta_n - z_1\sin\theta_n)+\cdots+(z_{m-1}\sin\theta_{n-1}+z_m\cos\theta_{n-1}) \end{array}\right))] \vee \\ &O(t_1,\ldots,t_m). \end{aligned} \tag{5.5}$$

As before, we can represent the C-space obstacle, as the result of a projection from the $2m+n$-dimensional space with coordinates $(z_1,\ldots,z_m,t_1,\ldots,t_m,\theta_1,\ldots,\theta_n)$ to $m+n$-dimensional C-space, with coordinates $(t_1,\ldots,t_m,\theta_1,\ldots,\theta_m)$, by using quantifier elimination on the above quantified representations. That is, we project out the variables $z_1,\ldots,z_m$.

S_3 is a subset of the C-space obstacle which also intersects S_1 but it is easy to compute. Additionally, this also works if instead of the boundary of the obstacle we use some approximation which contains all the boundaries of the obstacles and also is in the obstacle.

A possible advantage of representing C-space obstacle this way is that the bound-

ary of the obstacle is a union of line segments and curves if the obstacle is 2-dimensional.

Additionally, if the object may translate only in l dimensions, where $l < m$, the C-space obstacle in this case is:

$$\exists(z_1,\ldots,z_m)[P(z_1,\ldots,z_m) \wedge Boundary(O\begin{pmatrix} t_1 + (z_1\cos\theta_1 - z_2\sin\theta_1) + \cdots + (z_m\sin\theta_n + z_1\cos\theta_n),\ldots, \\ t_m + (z_m\cos\theta_n - z_1\sin\theta_n) + \cdots + (z_{m-1}\sin\theta_{n-1} + z_m\cos\theta_{n-1}) \end{pmatrix})] \vee \exists(z_1,\ldots,z_m)[O(t_1,\ldots,t_l,t_{l+1}+z_{l+1},\ldots,t_m+z_m)] \tag{5.6}$$

5.2 Quantifier-free C-space Obstacles

We have established that the C-space obstacle can be defined by a block of existential quantifiers applied to a condition on a higher dimensional space or the Omnispace.

In order to represent the C-space obstacles using only Boolean operators we could apply elimination of quantifiers to the C-space obstacles representation in the Omnispace. Once the bound variables are eliminated we obtain the C-space obstacle in lower dimension. Additionally, in the case of quantified representation of C-space obstacle with boundary operator, we also have to apply to boundary operator to the primitives.

However, due to the high complexity of the quantifier elimination algorithm, it may be desirable to use pruning and spatial subdivision with quantifier elimination.

5.2.1 Models and Boxes

In order to employ the pruning and subdivision technique using interval arithmetic, we use the above geometric notions of models and boxes developed in Chapter 3 to define the Omnimodel.

Let model M be a tree structure of extended semi-algebraic atomic formula with variable from $x_1, \ldots, x_n$ at the leaves and Boolean operators at the nodes. If B is an n-dimensional box defined by intervals, the pair (M, B) define the set of points in B that satisfy M.

For a single m-dimensional object translate and rotate in the presence of several m-dimensional static obstacles in the same m-dimensional Workspace, we assume that the object is connected and that it is much smaller that the Workspace. Also assume that the object is contained, tightly, in an m-dimensional box B_p.

It is clear that the dimensions of the C-space which correspond to the translational configuration of the object is m-dimensional. Thus we assume that it is bounded by an m-dimensional box. The size of this box is the same as the Workspace since the maximum space in which the reference point of the object may translate is the entire Workspace. Call this translation box B_t.

On the other hand, the dimensions of the C-space which correspond to the rotation configuration of the object is n-dimensional where $n = \frac{1}{2}m(m-1)$. We assume that this space is bounded by an n-dimensional box with the width in radians. Since the configuration of the object is the same at 0 and 2π, each side of this box can be limited to the range $[0, 2\pi)$ radians. Call this rotation box B_r

The Omnispace is a $(2m+n)$-dimensional space bounded by a box $B_p \times B_t \times B_r$ and the C-space is an $(m+n)$-dimensional space bounded by a box $B_t \times B_r$.

For example, consider the case of a ladder of two unit length and of infinitesimal width in an L-shaped corridor of a unit width in 2-dimensional space (Figure 5-1).

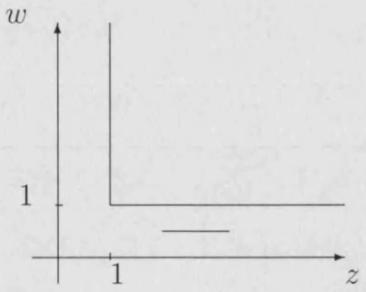


Figure 5-1: A ladder in an L-shape corridor.

The ladder can be defined by

$$P = \{(z, w) : z > 0 \ \wedge \ z < 2 \ \wedge \ w = 0\}$$

and the walls of the corridor by:

$$O = \{(z, w) : (z \geq 1 \ \wedge \ w \geq 1) \vee z \leq 0 \vee w \leq 0\}$$

If the ladder is allowed to translate in each dimension by x and y respectively, and rotate around its reference point by θ then C-space obstacles of the ladder due to the corridor can be defined as:

$$\begin{aligned} CO_P(O) \ = \ & \{(x, y, \theta) : \exists z \exists w [z > 0 \ \wedge \ z < 2 \ \wedge \ w = 0 \wedge \\ & ((z\cos(a) - w\sin(a) + x \geq 1 \ \wedge w\cos(a) + z\sin(a) + y \geq 1) \vee \\ & z\cos(a) - w\sin(a) + x \leq 0 \vee w\cos(a) + z\sin(a) + y \leq 0)]\} \end{aligned}$$

Assuming that the Workspace is limited to $0 \leq z \leq 4$ and $0 \leq w \leq 4$, we can bound the translation of the object by the box $B_t = [0, 4] \times [0, 4]$ whereas the ladder is contained in the box $B_p = [0, 2] \times [0, 1]$. Since the ladder can rotate in one dimension, this can be contained in the box $B_r = [0, 2\pi)$.

The Omnispace is the box $([0, 4] \times [0, 4] \times [0, 2] \times [0, 1] \times [0, 2\pi))$ and the C-space is the box $([0, 4] \times [0, 4] \times [0, 2\pi))$.

5.2.2 Using Pruning and Subdivision

We explore the possibility of using a spatial subdivision technique along with elimination of quantifiers to speed-up the calculation of quantifier-free C-space obstacles. This is based on the assumption that the objects and their obstacles are usually globally complicated but locally simple and that elimination of quantifiers has computational difficulty which increases much more than linearly with the complexity of formulae.

We use coordinate-aligned boxes division since it is intuitively simple. It is also simple to perform subdivision techniques such as grid-divisions and recursive subdivision, on coordinate-aligned boxes.

For n-dimensional boxes which are defined by n closed intervals, grid-division can be done on selected dimensions and the original box is replaced by a grid of boxes. The original model is then get pruned to each box. In contrast, recursive subdivision is done on one selected dimension at a time. The process divides the longest side of the box and the original box is replaced by two adjacent boxes. The original model is then get pruned to both boxes and the process continues until some conditions are met. With recursive subdivision, boxes are subdivided only when and where it is necessary.

Subdivision Considerations

C-space obstacle is the result of applying quantifier elimination to the Omnimodel. We can subdivide the Omnispace and prune the Omnimodel to each subspaces. The desirable outcome is that each subspace will have less complicated Omnimodel inside.

In our case the Omnispace is the box $B_p \times B_t \times B_r$. The box B_p, which corresponds to the bound variable, is generally small and fit the object snuggly therefore we need not subdivide sides of the box which correspond to bound variables.

5.2.3 Limitations

Whether or not we can use the quantifier elimination to obtain the quantifier-free representation of C-space obstacle depends on the representation of C-space obstacle in quantified form. The elimination of existential quantifier from extended semi-algebraic sets are only guaranteed to be semi-algebraic if the variables being projected only occur algebraically. In order to represent the C-space obstacle with Boolean operator alone by using quantifier elimination, the existential quantifier in the representation of the C-space obstacle need to treat the elementary functions *sine*, *cosine* as parameters. With models consisting of atomic formulae from extended semi-algebraic sets, if we wish to eliminate quantifiers then the bound variables cannot appear in *sine*, *cosine* or *exp* function.

5.3 Computational Experiments

The simplest type of spatial planning problems are those that concerned with a single moving convex object among known static convex obstacles. In this thesis, we consider several example cases of a single 2-dimensional object avoiding several 2-dimensional static obstacles in a 2-dimensional Workspace. The dimensions of the Omnispace is between 4 and 6, depending on the degrees of freedom of the object and the resulting C-space, after the elimination of quantifier, is at most 3-dimensional. Additionally, the object and the obstacles are not limited to convex polygons, they may be non-convex and have curved edges.

5.3.1 Software Used and Developed

The experiment in this thesis was conducted using computer logic system REDLOG 2.0 implemented in REDUCE computer algebra system.

REDUCE is a powerful Computer Algebra system available for many operating systems. The elementary functions are easy to use and for non-interactive mode

users can write new procedures using REDUCE syntax which is PASCAL-like. The system was described in more details in, for example [28] [44].

REDLOG [20] [22], which stands for REDuce LOGic system, provides an extension to REDUCE computer algebra system so that logical expressions and quantifiers can be dealt with. It provides many functions for the symbolic manipulation of first order formula over some temporarily fixed languages and theories. The focus of the system is on simplification of quantifier-free formula and effective quantifier elimination [21].

The algorithms implemented as REDLOG functions include:

- Several techniques for the simplification of quantifier-free formula.
- Quantifier elimination.
- Linear optimisation using quantifier elimination techniques.
- CNF/DNF computation.

Limitations

The implementation of quantifier elimination functions in REDLOG is limited to at most quadratic occurrences of quantified variables. The CPU time for each particular case is limited to 15 minutes.

Implementation of Algorithms

We developed a system of REDUCE procedures using REDUCE syntax to implement our algorithms and conduct experiments. The procedures are relatively high-level which allow input of the form:

```
procedure_name(object,
```

```
obstacle,
box,
number_of_bound_varialble,
variable);
```

where `object` and `obstacle` are semi-algebraic descriptions, `box` is a list of intervals, `number_of_bound_variable` is an integer and `variable` is a list of variables available to `object` and `obstacle`.

The test is performed a number of times. We measure the CPU-time taken to perform the calculation using the timing functions provided by REDUCE. The average of these times is taken, as the time taken to calculate the example problems. The timing function which return the time in milliseconds, is system and implementation dependent. The system used in our implementation was a single processor Intel Pentium III 500 MHz with 128Mb of memory. The operating system was RedHat Linux 6.1.

In many cases, the results were omitted since the output in non-quantified form cannot be obtained. We denote the case where the calculation was not completed due to the CPU-time required exceeded the limit by '??' and denote the case where the result cannot be obtained due to the limitation of the REDLOG quantifier elimination procedures by '–'.

5.3.2 Test Problems

Variable List

Let (z, w, x, y, a) be the variable list available for both the object and obstacles throughout the experiment.

Degrees of Freedom

We consider three cases of different degrees of freedom of a 2-dimensional moving object. These are:

One degree of freedom Object may translate in one dimension but may not rotate.

Two degrees of freedom Object may translate in both dimensions but may not rotate.

Three degrees of freedom Object may translate in both dimensions and may rotate about the reference point.

Boxes

For each case in the experiment the bounding boxes used are:

	DoF		
	1	2	3
Box B_p for P_1	$[0,1] \times [0,1]$	$[0,1] \times [0,1]$	$[0,1] \times [0,1]$
Box B_p for P_2	$[0,2] \times [0,3]$	$[0,2] \times [0,3]$	$[0,2] \times [0,3]$
Box B_p for P_3	$[-1,1] \times [-2,2]$	$[-1,1] \times [-2,2]$	$[-1,1] \times [-2,2]$
Box B_p for P_4	$[-1,1] \times [-1,1]$	$[-1,1] \times [-1,1]$	$[-1,1] \times [-1,1]$
Translation box B_t	$[0,64]$	$[0,64] \times [0,16]$	$[0,64] \times [0,16]$
Rotation box B_r	–	–	$[0,7]$

For example, the Omnispace of the object P_2 which has one degree of freedom and the object O_3 is the box $[0,2] \times [0,3] \times [0,64]$, after the quantifier elimination the resulting C-space obstacle is bounded by a box $[0,64]$, whereas the Omnispace of the object P_3 which has 3 degree of freedom and the object O_1 is the box $[-1,1] \times [-2,2] \times [0,64] \times [0,16] \times [0,7]$ and after the quantifier elimination the resulting C-space obstacle is bounded by a box $[0,64] \times [0,16] \times [0,7]$.

Objects

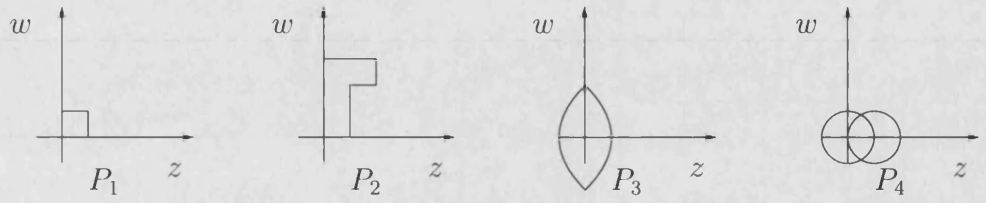


Figure 5-2: 2-dimensional movable objects.

We also consider four different objects varying in their complexity (Figure 5-2). They are also differ in their convexity and by different degrees of polynomials of the semi-algebraic atomic formulae.

P_1 is a convex object defined by a combination of four atomic formulae of degree one.

$$P_1 = intersection(z \geq 0, z \leq 1, w \geq 0, w \leq 2)$$

P_2 is a non-convex object defined by a combination of eight atomic formulae of degree one.

$$\begin{aligned} P_2 \quad = union(\quad & intersection(z \geq 0, z \leq 1, w \geq 0, w \leq 3), \\ & intersection(z \geq 1, z \leq 2, w \geq 2, w \leq 3)) \end{aligned}$$

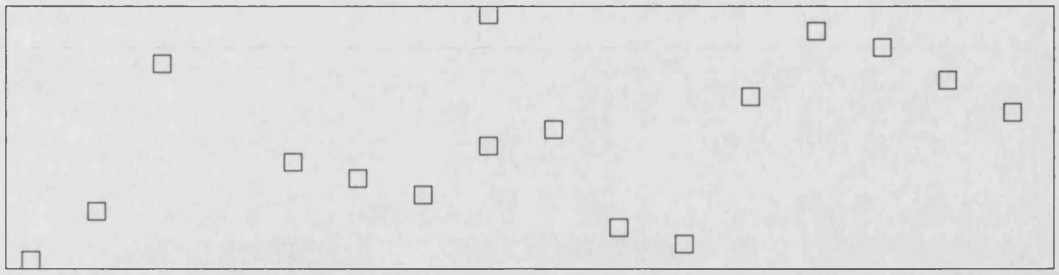
P_3 is a convex object defined by an atomic formula of degree two.

$$P_3 = z^2 + \frac{1}{4}w^2 - 1 \leq 0$$

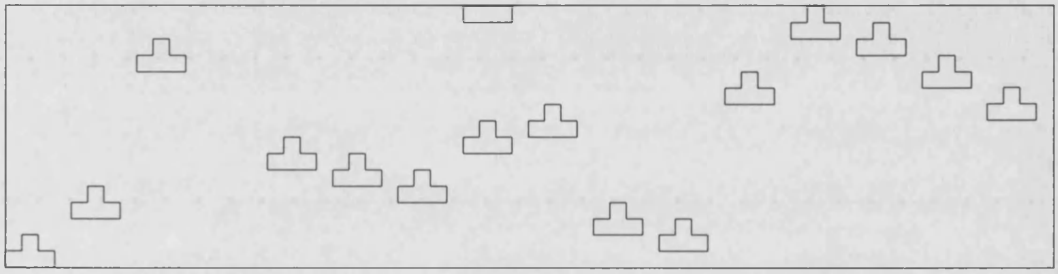
P_4 is a non-convex object defined by a combination of two atomic formulae of degree two.

$$P_4 = intersection(z^2 + \frac{1}{4}w^2 - 1 \leq 0, z^2 + \frac{1}{4}(w-1)^2 - 1 \leq 0)$$

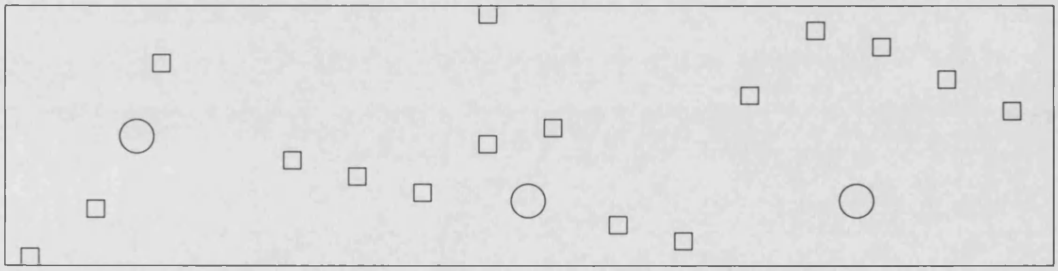
Obstacles



O_1



O_2



O_3

Figure 5-3: Sets of 2-dimensional obstacles.

We consider three different sets of obstacles varying in their complexity (Figure 5-3). They are also differ in their convexity and by different degrees of polynomials of the semi-algebraic atomic formulae.

O_1 is a set of 16 convex objects, each defined by a combination of four atomic formulae of degree one.

O_2 is a set of 16 non-convex object, each defined by a combination of eight atomic formulae of degree one.

O_3 is a set of convex object, 3 of which are defined by an atomic formula of degree two and 16 of which are defined by a combination of four atomic formulae of degree one.

5.3.3 C-space Obstacles Representation

The main concern of this experiment is the comparison between two C-space representations.

Results

We present results of using quantifier elimination procedures provided by REDLOG on two different C-space obstacles representations: the representation which uses existential quantifiers (Q) and the representation which uses existential quantifiers and boundary formations (Q & B). (See Table 5.1)

We examine the time taken to compute the quantifier-free C-space obstacles and the size of the quantifier-free output.

It is clear that representing C-space obstacles using only existential quantifiers resulted in faster computing time and generally more compact output than incorporating the boundary formation. This may due to the fact that the naive boundary formation of the obstacles leads to an increase in the number of the input formulae.

Discussions

There are two important considerations, the time for the computation and the size of the output. Both of these vary in an unstable way depending on the precise

		Time		No. of Formulae	
		Q	Q & B	Q	Q & B
1 DoF	P_1O_1	0.2	0.4	9	7
	P_1O_2	0.2	0.8	13	15
	P_1O_3	0.4	1.4	26	32
	P_2O_1	0.2	0.6	27	25
	P_2O_2	0.3	1.0	42	42
	P_2O_3	0.6	1.8	71	77
	P_3O_1	0.4	1.5	64	79
	P_3O_2	0.8	3.3	120	155
	P_3O_3	–	–	–	–
	P_4O_1	0.3	1.7	35	48
	P_4O_2	0.5	3.8	79	113
	P_4O_3	10.6	103.7	1304	5501
2 DoF	P_1O_1	1.0	3.0	315	424
	P_1O_2	2.0	6.3	704	916
	P_1O_3	3.4	10.0	1183	1491
	P_2O_1	1.0	2.9	418	522
	P_2O_2	2.1	6.3	883	1106
	P_2O_3	3.7	10.3	1515	1860
	P_3O_1	15.0	58.7	5629	10179
	P_3O_2	36.5	128.0	11208	18200
	P_3O_3	–	–	–	–
	P_4O_1	24.0	86.6	8067	15584
	P_4O_2	59.7	194.9	16371	27811
	P_4O_3	578.3	??	35093	60398
3 DoF	P_1O_1	29.3	69.7	3852	3901
	P_1O_2	70.1	161.0	6942	7009
	P_1O_3	–	–	–	–
	P_2O_1	35.7	76.8	4775	4824
	P_2O_2	83.1	176.0	8541	8608
	P_2O_3	–	–	–	–
	P_3O_1	??	??	??	??
	P_3O_2	??	??	??	??
	P_3O_3	–	–	–	–
	P_4O_1	??	??	??	??
	P_4O_2	??	??	??	??
	P_4O_3	–	–	–	–

Table 5.1: Computing Time (seconds) and number of atomic formulae of C-space obstacles.

formulation of the problem.

However, we expected that the boundary formulation of the C-space obstacle would be better than the basic formulation which uses existential quantifier alone, since the formulae is of lower dimension. We have not seen this improvement. It may be that we do not yet know how to compute and represent boundary of obstacles efficiently. We are certainly not ready to abandon this approach.

5.3.4 Subdivision Algorithms

We continue using quantifier elimination procedures provided by REDLOG but incorporate our pruning and subdivisions procedures. We consider two types of subdivision algorithms: the grid divisions and recursive subdivision. Moreover, for each subdivision algorithm we consider four different objects with varying complexity moving among three different sets of obstacles as in the previous example.

For the grid division technique, we consider 4 different grid sizes: 8, 16, 32 and 64 units. Similarly, for the recursive subdivision, the termination conditions of the algorithm are also based on the size of boxes. Although we stated in the Algorithm 3, we do not consider the case of $IsSimple()$. We also consider the 4 sizes: 8, 16, 32 and 64 units.

For both subdivision technique, the size 64 implies that subdivisions do not occur though we perform the pruning process on the model tree. The time reported are the total time of computing every sub-boxes. The number of atomic formulae are also the total number from every sub-boxes.

Results

The results of using quantifier elimination with pruning and subdivisions as a pre-process are as follow:

Table 5.2 5.4 and 5.6 shows the computing time used to compute quantifier-free C-space obstacles and the Table 5.3 5.5 and 5.7 shows the number of atomic formulae of quantifier-free C-space obstacles for both division techniques.

Although recursive subdivisions seem to allow faster computation than the grid division the subdivision process does not improve the speed of the calculation. However, as the complexity of the problem increases both subdivision technique begin to improve the speed of the calculations. In some cases, the output can only be obtained in within the limited CPU time by using subdivisions.

The number of atomic formulae by both subdivision techniques are naturally the same throughout. However, in some circumstances the subdivision process reduces the number of atomic formulae of the output.

Discussions

The main concern of this experiment is to gain an insight into how to use pruning and division technique as a pre-process to the quantifier eliminations. Algebraically, we are eliminating 2 variables from the total of 3 to 5 variables.

We have described and experiment two subdivisions techniques. The examples illustrate the methods but it does not seem possible to come to any definite conclusions about these methods at this point but we can make a few remarks.

The time for the computation vary in a quite unstable way depending on the subdivision technique, and also on the maximum size of of the boxes. We believe in general that spatial subdivision and pruning, if done correctly, can be useful as pre-processing to quantifier elimination. In one of our examples, the quantifier elimination only succeeded if it was preceded by spatial subdivision and pruning.

		Grid				Recursive			
		8	16	32	64	8	16	32	64
1 DoF	P_1O_1	17.3	4.4	2.2	1.1	2.3	1.5	1.3	1.1
	P_1O_2	33.4	8.4	4.2	2.2	3.9	2.8	2.5	2.2
	P_1O_3	36.1	9.1	4.7	2.4	5.3	3.4	2.9	2.5
	P_2O_1	18.4	4.7	2.4	1.3	4.5	2.4	1.8	1.3
	P_2O_2	34.7	8.7	4.5	2.4	7.2	4.1	3.2	2.4
	P_2O_3	37.9	9.7	5.1	2.8	10.5	5.2	3.9	2.8
	P_3O_1	18.0	4.6	2.5	1.5	4.6	2.4	1.9	1.5
	P_3O_2	34.7	8.9	4.8	2.7	7.2	4.1	3.4	2.8
	P_3O_3	–	–	–	–	–	–	–	–
	P_4O_1	18.8	4.8	2.5	1.4	5.2	2.6	1.9	1.4
	P_4O_2	35.1	8.9	4.6	2.5	7.5	4.1	3.2	2.6
	P_4O_3	39.2	10.3	6.1	4.2	10.9	5.4	4.6	4.2

Table 5.2: Computing Time (seconds) of 1-dimensional C-space obstacles.

		Grid				Recursive			
		8	16	32	64	8	16	32	64
1 DoF	P_1O_1	12	6	6	7	12	6	6	7
	P_1O_2	12	10	10	11	12	10	10	11
	P_1O_3	16	12	13	14	16	12	13	14
	P_2O_1	40	18	18	25	40	18	18	25
	P_2O_2	56	31	31	40	56	31	31	40
	P_2O_3	110	55	55	62	110	55	55	62
	P_3O_1	156	62	62	64	156	62	62	64
	P_3O_2	298	123	123	120	298	123	123	120
	P_3O_3	–	–	–	–	–	–	–	–
	P_4O_1	50	23	23	35	50	23	23	35
	P_4O_2	126	68	68	79	126	68	68	79
	P_4O_3	282	146	302	352	282	146	302	352

Table 5.3: Number of atomic formulae of 1-dimensional C-space obstacles.

		Grid				Recursive			
		8	16	32	64	8	16	32	64
2 DoF	P_1O_1	22.4	6.1	3.5	2.3	13.9	7.5	4.8	2.4
	P_1O_2	42.8	11.9	6.9	4.8	25.8	14.3	9.5	4.8
	P_1O_3	47.9	14.1	8.6	6.6	30.9	17.2	11.7	6.7
	P_2O_1	23.9	6.5	3.7	2.5	16.1	7.9	5.0	2.5
	P_2O_2	44.6	12.5	7.2	5.0	29.6	15.2	10.0	5.1
	P_2O_3	50.7	15.7	9.6	7.5	35.3	18.6	12.5	7.2
	P_3O_1	35.2	15.4	14.4	18.0	24.5	15.7	14.9	16.7
	P_3O_2	73.5	35.3	33.5	40.5	46.5	33.3	33.1	37.6
	P_3O_3	–	–	–	–	–	–	–	–
	P_4O_1	39.6	20.5	21.3	28.1	29.7	20.7	21.2	26.2
	P_4O_2	81.4	46.1	48.7	63.2	54.3	44.1	47.4	59.1
	P_4O_3	??	616.2	699.2	908.6	296.0	379.5	??	567.3

Table 5.4: Computing Time (seconds) of 2-dimensional C-space obstacles.

		Grid				Recursive			
		8	16	32	64	8	16	32	64
2 DoF	P_1O_1	304	286	304	360	304	286	304	360
	P_1O_2	565	612	650	696	565	612	650	696
	P_1O_3	1000	1049	1053	1154	1000	1049	1053	1154
	P_2O_1	404	339	375	414	404	339	375	414
	P_2O_2	828	755	807	879	828	755	807	879
	P_2O_3	1508	1397	1375	1511	1508	1397	1375	1511
	P_3O_1	4680	4354	4849	5629	4680	4354	4849	5629
	P_3O_2	9596	9282	10093	10768	9596	9282	10093	10768
	P_3O_3	–	–	–	–	–	–	–	–
	P_4O_1	5508	5910	6774	7977	5508	5910	6774	7977
	P_4O_2	11466	12706	13997	15392	11466	12706	13997	15392
	P_4O_3	22760	26717	30096	33580	22760	26717	30096	33580

Table 5.5: Number of atomic formulae of 2-dimensional C-space obstacles.

		Grid				Recursive			
		8	16	32	64	8	16	32	64
3 DoF	P_1O_1	107.9	41.7	33.7	35.0	69.3	41.3	34.9	33.1
	P_1O_2	236.7	94.6	77.2	81.2	141.0	88.2	76.7	76.6
	P_1O_3	816.7	–	–	–	629.1	–	–	–
	P_2O_1	202.3	63.3	46.7	42.8	141.2	61.4	47.4	40.4
	P_2O_2	??	150.0	111.1	101.5	291.2	134.9	106.2	92.5
	P_2O_3	??	–	–	–	??	–	–	–
	P_3O_1	??	??	??	??	??	??	??	??
	P_3O_2	??	??	??	??	??	??	??	??
	P_3O_3	–	–	–	–	–	–	–	–
	P_4O_1	??	??	??	??	??	??	??	??
	P_4O_2	??	??	??	??	??	??	??	??
	P_4O_3	??	??	??	??	??	??	??	??

Table 5.6: Computing Time (seconds) of 3-dimensional C-space obstacles.

		Grid				Recursive			
		8	16	32	64	8	16	32	64
3 DoF	P_1O_1	7953	5040	4324	3852	7953	5040	4324	3852
	P_1O_2	14466	8985	7672	6800	14466	8985	7672	6800
	P_1O_3	18504	11641	9723	8652	18504	11641	9723	8652
	P_2O_1	18988	7719	5838	4775	18988	7719	5838	4775
	P_2O_2	??	13987	10618	8541	33312	13987	10618	8541
	P_2O_3	??	–	–	–	??	–	–	–
	P_3O_1	??	??	??	??	??	??	??	??
	P_3O_2	??	??	??	??	??	??	??	??
	P_3O_3	–	–	–	–	–	–	–	–
	P_4O_1	??	??	??	??	??	??	??	??
	P_4O_2	??	??	??	??	??	??	??	??
	P_4O_3	??	??	??	??	??	??	??	??

Table 5.7: Number of atomic formulae of 3-dimensional C-space obstacles.

5.3.5 Complexity Estimates

As mention in Section 4.2 of Chapter 4 that eliminating a block of n_0 existential quantifiers has complexity bounded approximately by

$$(md)^{n_0 n_1 C} Cost$$

where C is some constants, n_1 is the number of free variables and d is the maximum degree of m different polynomials involved. We expect complexity for our problem to increase as some power of md.

The experimental results above showed that in many cases, pruning and subdivisions of boxes, in some circumstances, may leads to a smaller value of m and d.

5.3.6 Limitations

In practice, is does not seem possible to compute quantifier-free C-space beyond linear case with a few variables. Quantifier elimination is computationally expensive and there is an evidence that many sets which have a succinct representation with the projection operator are impossible to represent in any model of feasible size which only uses Boolean operators [65]..

Additionally, it is only possible in principle, to replace the quantified formulae with quantifier-free ones if the bound variables appear only algebraically. Even when it is possible to get rid of the projection operator using quantifier elimination, it is by no means clear that it is desirable to do so.

Alternatively, we may consider working directly with the extended CSG representations by explore the possibility of using pruning and recursive subdivision directly with extended CSG representation.

5.4 Summary

We presented methods to construct C-space obstacle using existential quantifiers and boundary formation. The quantified C-space obstacle represents a geometric object in the Omnispace. We can obtain the quantifier-free C-space obstacle by applying quantifier eliminations.

We introduced the applications of pruning and spatial subdivision techniques to speed-up the computation of quantifier-free representation. We presented some experimental results and discussed the effectiveness of the methods.

Chapter 6

Extended CSG System

In this chapter we introduce an extended version of semi-algebraic CSG which has projection and boundary as operators, as well as the usual Boolean ones. We suggest how this idea can be applied to spatial planning by representing C-space obstacle using primitives from this system. We also suggest how to partially solve Findspace and Findpath problem using this system.

6.1 Extended Operators

Whether or not we extend the semi-algebraic primitives to include trigonometric and exponential functions, we may wish to extend the set of operators. We now wish to add two new operators: boundary and projection when we represent complex objects using extended semi-algebraic primitives.

6.1.1 Boundary Operator

Definition 6 *Let S be a subset of R^n. Define $Closure(S)$ to be the set of points which are either in S or which have elements of S arbitrarily near them.*

For example, if $S = \{x : x < 0\}$ then $Closure(S) = \{x : x \leq 0\}$ whereas if $S = \{x : x = 1\}$ then $Closure(S) = \phi$.

Definition 7 *Let S be a subset of R^n. Define*

$$Boundary(S) = Closure(S) \cap Closure(\overline{S})$$

where $\overline{S} = R^n - S$. A point is in $Boundary(S)$ if it is arbitrarily close to points in S and also close to points in $\overline{S}$.

However, since set complement operator depends on the universe, so does the boundary operator. If the universe is B and $S \subseteq B$ then

$$Boundary(S) = Closure(S) \cap Closure(B - S).$$

Additionally, if S is a semi-algebraic set then $Closure(S)$ and $Boundary(S)$ are semi-algebraic.

For example, for $S_1, S_2 \subseteq R^2$ the set $S_1 = \{(x, y) : x^2 + y^2 - 1 \leq 0\}$ defines a disc and $Boundary(S_1) = \{(x, y) : x^2 + y^2 - 1 = 0\}$ defines the circle in R^2. Similarly, the set $S_2 = \{(x, y) : y \leq x^2\}$ defines a region below the parabola and $Boundary(S_2) = \{(x, y) : y = x^2\}$ defines the parabola in R^2.

We wish to add the boundary operator to our list of possible operators.

6.1.2 Projection Operator

Define the projection operator as follows:

Definition 8 *Let S be a subset of R^n. Suppose $V = \{y_1, \ldots, y_k\}$ is a subset of the variables $\{x_1, \ldots, x_n\}$. Let $W = \{w_1, \ldots, w_j\}$ be the variables in $\{x_1, \ldots, x_n\}$ but not in V.*

Define a projection of variables in V of set S as:

$$Projection_V(S) = \{(w_1, \ldots, w_j) : (x_1, \ldots, x_n) \in S\}$$

for some $\{y_1, \ldots, y_k\}$.

Define cylindrical projection of variable in V of set S as:

$$CylProjection_V(S) = \{(x_1, \ldots, x_n) : (w_1, \ldots, w_j) \in Projection_V(S)\}$$

Both projection operators have two arguments, a subset of R^n and a set of variables to be projected out. The interpretation of *Projection* is as the usual projection operator where the result of the projection is in the space of the remaining variables. On the other hand, the interpretation of *CylProjection* projects the specified variables in the set S and leave the result in the same space. The result of this projection is regarded as "cylindrical". In other words, $Projection_V(S)$ is a subset of R^k obtained by taking all points $(x_1, \ldots, x_n)$ in S and forgetting all coordinates of variables in V. In contrast, $CylProjection_V(S)$ is the set of points $(x_1, \ldots, x_n)$ in R^n which can be transformed into points of S by changing values of variables in V. Therefore $CylProjection_V(S)$ is a cylinder in R^n whose base is $Projection_V(S)$.

For example, the set $S = \{(x, y, z) : x^2 + y^2 + z^2 - 1 \leq 0\}$ defines a sphere. From the definition above

$$Projection_{\{z\}}(S) = \{(x, y) : x^2 + y^2 - 1 \leq 0\}$$

is a disc in xy-space, whereas

$$CylProjection_{\{z\}} = \{(x, y, z) : x^2 + y^2 - 1 \leq 0\}$$

is a cylinder in xyz-space which has the disc $Projection_{\{z\}}(S)$ as a base.

6.1.3 Relationship Between Operators

Subsets of R^n obtained from extended semi-algebraic primitives using combinations of Boolean operators; n-ary union, n-ary intersection, unary complement, and the two new operators; unary boundary and unary projection, provide finite description of geometric objects and a set of operators capable of manipulating them. Call this system *Extended CSG System.*

The original CSG system is a Boolean algebra of extended semi-algebraic sets therefore CSG models is stable since the Boolean operators are supported by well-defined set of axiom. In contrast, we do not at present know how to compute with this Extended CSG system.

However, we can establish some relationship between the two new operators and the usual Boolean operators as follow:

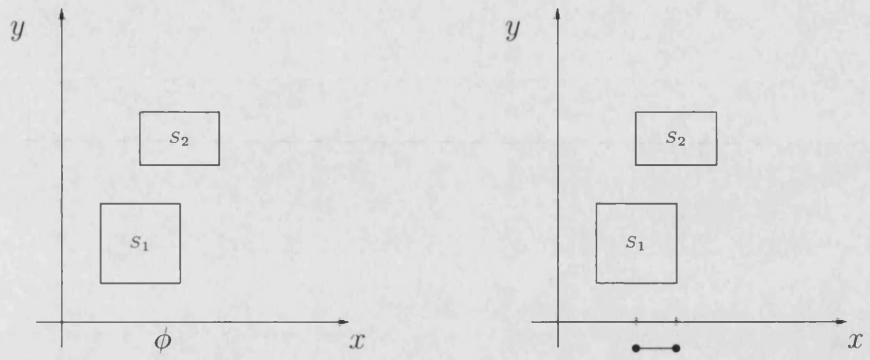
- $Projection_{\{V\}}(S_1 \cup S_2)) = Projection_{\{V\}}(S_1) \cup Projection_{\{V\}}(S_2)$
- $CylProjection_{\{V\}}(S_1 \cup S_2)) = CylProjection_{\{V\}}(S_1) \cup CylProjection_{\{V\}}(S_2)$
- $Boundary(S) = Boundary(complement(S))$
- $Boundary(\phi) = Boundary(universe) = \phi$
- $Boundary(S_1 \cup S_2)) \subseteq (Boundary(S_1) \cup Boundary(S_2))$
- $Boundary(S_1 \cap S_2) \subseteq (Boundary(S_1) \cap Boundary(S_2))$

Although the above expressions appear to be true, similar expressions are not. For example, as illustrated in Figure 6-1.

$$Projection_{\{y\}}(S_1 \cap S_2) \neq Projection_{\{y\}}(S_1) \cap Projection_{\{y\}}(S_2)$$

Similarly,

- $Projection_V(S_1 \cap S_2) \neq Projection(S_1) \cap Projection(S_2)$



$Projection_{\{y\}}(S_1 \cap S_2)$ $Projection_{\{y\}}(S_1) \cap Projection_{\{y\}}(S_2)$

Figure 6-1: $Projection_{\{y\}}(S_1 \cap S_2) \neq Projection_{\{y\}}(S_1) \cap Projection_{\{y\}}(S_2)$

- $Boundary(S_1 \cup S_2) \neq Boundary(S_1) \cup Boundary(S_2)$
- $Boundary(S_1 \cap S_2) \neq Boundary(S_1) \cap Boundary(S_2)$

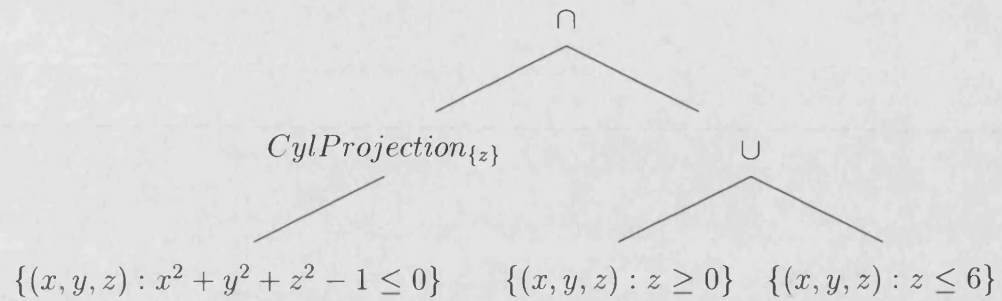
6.2 Model and Boxes

As in the case of CSG system, description of objects in Extended CSG system can be represented by a data structure: a tree with operators on the internal nodes and extended semi-algebraic primitives at the leaves.

For example, an Extended CSG object defined by

$$CylProjection_{\{z\}}(x^2 + y^2 + z^2 - 1 \leq 0) \cap (\{(x, y, z) : z \geq 0\} \cup \{(x, y, z) : z \leq 6\})$$

can be represented as:



With these extra operators and primitives, we can represent geometric objects which are defined by projections and boundary formation in a succinct form. How to compute with these such representation is our *Basic Problem* which includes how to do the pruning on the model consisting of primitives from this extended CSG.

In order to use the pruning and subdivision technique on the models of Extended CSG system, we use the idea of models and boxes introduced in Chapter 5.

A model M is represented by a tree with atomic formulae with variable from $(x_1, \ldots, x_n)$ at the leaves and geometric operators at the internal nodes. The atomic formulae in M are those of extended semi-algebraic primitives. The geometric operators are n-ary union, n-ary intersection, unary complement, unary boundary and unary projection all of which operate on sets.

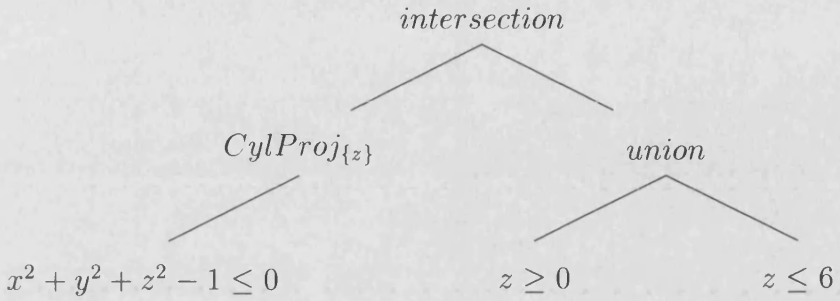
Unlike extended CSG object tree which defines a subset of R^n, a model trees such as M describe conditions of variable $(x_1, \ldots, x_n)$ available to them. We distinguish between the two types of tree structure by using the following notations:

	Extended CSG Tree	Model Tree
Primitives	Semi-algebraic Sets	Atomic Formulae
Operators	$\cup$	*union*
	$\cap$	*intersection*
	$-$	*complement*
	Boundary	*Bnd*
	$Projection_V$	$Proj_V$ †
	$CylProjection_V$	$CylProj_V$

For example, for a variable list (x_1, x_2, x_3) a model

$$M = intersection(CylProj_{\{z\}}(x^2 + y^2 + z^2 - 1 \leq 0), union(z \geq 0, z \leq 6))$$

can be represented as:



Let B be an m-dimensional coordinate-aligned box defined by a list of m closed intervals $([a_1, b_1], \ldots, [a_m, b_m])$ where $m \leq n$. Each interval corresponds to an edge of the box. The set (M, B) is a set of points in B that satisfy M. This set definition is recursive on the structure of M. If M is an atomic formula then (M, B) will be the subset of B in which M is *true* and it follows that:

- $(union(M_1, M_2), B) = (M_1, B) \cup (M_2, B)$
- $(intersection(M_1, M_2), B) = (M_1, B) \cap (M_2, B)$
- $(complement(M), B) = B - (M, B)$
- $(Bnd(M), B) = Closure(M, B) \cap Closure(B - (M, B))$
- $(CylProj_V(M), B) = CylProjection_V(M, B)$

We will use a pair (M, B), where M is a model and B is a box, to define a subset of R^n. This set (M, B) is our primitive Extended CSG object. For example, let B be the box $([-10, 10], [-10, 10])$. Let (x, y) be the variable list. That is B is a box in xy-space. Let M be a model defined by $y \leq x^2$. (M, B) defines the region inside the box and below the parabola, where M is *true*. $Boundary(M, B)$ is the parabola inside the box.

[†]In this thesis, we only allow the cylindrical interpretation for M.

Additionally, suppose the ordered list of variables is $x_1, \ldots, x_n$ and a box B of closed intervals is $([a_1, b_1], \ldots, [a_m, b_m])$ where $m \leq n$. Since, the set (M, B) is a collection of points in B where M is true, $(true, B)$ is the box B itself and $(false, B)$ is the empty set.

6.2.1 Projection of Boxes

In extended CSG representation, a model M is a tree. We work always in subsets of the box B since we do not wish to change space every time we climb up or down the tree. Consequently, for M, the only interpretation of projection is as cylindrical projection.

In some circumstances where it is possible, we may allow the usual interpretation of projection operator. This can be done by defining $Proj_V(M, B) = Projection_V(M) \cap Projection_V(B)$ where $Projection_V(B)$ is a box which has intervals corresponding to variables in the variables list which are not in V, where B is an m-dimensional box and the corresponding variables list is $(x_1, \ldots, x_m)$.

For example, Let M be a model $x_1^2 + x_2^2 + x_3^2 - 1 \leq 0$. Suppose the variable list is (x, y, z) and a box $B = ([-8, 8], [-9, 9], [-10, 10])$. That is, (M, B) defines a sphere in R^3. The usual interpretation of projection $Proj_{\{x_3\}}(M, B) = Projection_{\{x_3\}}(M) \cap Projection_{\{x_3\}}(B)$ will give a disc in x_1, x_2-space whereas the cylindrical interpretation gives a cylinder in x_1, x_2, x_3-space which has the disc as a base.

6.3 Basic Problem

6.3.1 Evaluation Process

The evaluation process, which determines the value of each primitive over the box, is done by using interval arithmetic on the atomic formula of each primitives.

In Extended CSG system, the evaluation process remain the same as in the CSG system since it does concern with the operator of the primitives. The process still able to replace some primitives in the box by *true* or $false$ in the same manner as in the CSG system. That is, the primitive can be evaluated over the box according to Table 3.1 in Section 3.4.2.

The evaluation process may be able to replace some unnecessary formulae of the model in the box which can lead to a reduction of the number of formulae that made up a model. Once each formula in the model has been evaluated and some replaced with *true* or $false$, we can work up the model tree applying the operators to each leaf and achieve some simplification. Hence we need to know how to apply the two new operators to the model tree.

Simplification Rules for Boundary Operator

Since we interpret the boundary operator to the box B, we ignore any point outside B. When a formula evaluated to *true* it can be interpreted as M is *true* inside B. That is, the intersection of the model and the box is the box itself. It is clear that in this case the box B does not intersect the boundary of the set so we can replace the formula with $false$. Similarly, when the formula evaluated to $false$, no part of the set is inside B which also means that the boundary of the set does not intersect B and we can replace M with $false$. The original primitive is returned if the evaluation result is *undecided.*

Thus the simplification rules of the boundary operators are:

1. $(Bnd(true), B) = false$
2. $(Bnd(false), B) = false$
3. $(Bnd(undecided), B) = undecided.$

Simplification Rules for Projection Operator

In the case of projection operator there are two possibilities.

In the case of cylindrical projection, the model remains in the same space. In other words, the projected model and the original box B defines the set. Therefore, when M evaluated to *true*, every point inside the box B is *true*. The geometric interpretation is that, B is completely inside the model M. It is clear that the cylindrical projection of M will also contain B so we can replace M with *true*. On the other hand, when M evaluated to $false$, no points in side the box is *true* for M. That is, no part of M is inside the box B therefore the cylindrical projection of M does not intersect the box and we can replace it with $false$. The original model is returned if the evaluation result is *undecided*.

Thus the simplification rules for the cylindrical interpretation of projection operators are:

1. $CylProj_V(true, B) = (true, B) = true$
2. $CylProj_V(false, B) = (false, B) = false$
3. $CylProj_V(M, B) = (undecided, B) = undecided.$

In the case of the usual projection operator, the variable of the box is also projected. That is, the projection of the model goes into a projected box. Thus simplification rules for the usual interpretation of projection operators are:

1. $Proj_V(true, B) = (true, Proj_V(B)) = true$
2. $Proj_V(false, B) = (false, Proj_V(B)) = false$
3. $Proj_V(M, B) = (undecided, Proj_V(B)) = undecided.$

Simplification Rules for Extended CSG Operator

Pruning process ensure that, inside the box there are no unnecessary atomic formulae left in the model while the simplified model and the box is still representing the original set. We can simplify the model over the box by applying the rules of the operators to the atomic formulae of the model after the evaluation process.

The simplification rules for Extended CSG operators are:

- $undecided \cup true = true$
- $undecided \cup false = undecided$
- $undecided \cap true = undecided$
- $undecided \cap false = false$
- $Complement(true) = false$
- $Complement(false) = true$
- $Complement(undecided) = undecided$
- $Boundary(true) = false$
- $Boundary(false) = false$
- $Boundary(undecided) = undecided$
- $CylProjection_V(true) = true$
- $CylProjection_V(false) = false$
- $CylProjection_V(undecided) = undecided$

6.3.2 Pruning

The pruning process can be done in Extended CSG in a similar manner as in CSG. By evaluating each atomic formula at the leaf of the model tree, we can simplify the model over the box by applying the rules of the operators to the primitives and achieve some simplification. However, in the case of Extended CSG, the operators include the boundary and projection thus the simplification rules need to be carefully considered.

Boundary and projection operators can be dealt with in two ways. One way is to apply the operator to the models then use quantifier elimination, leaving the model with only Boolean operators. In the case of boundary operator, the result is a semi-algebraic form of the boundary of the model. Similarly for projection operator, by projecting out some variables, the result is a semi-algebraic form of the model with only Boolean operators. In this way, we are transforming an Extended CSG model to a CSG model.

The other way is to apply the pruning process without applying the boundary or projection of the models but proceed to evaluate each atomic formula at the leaf of the model over the box first. The result from the evaluation could be either *true*, $false$ or *undecided*. Once each atomic formula in the model has been evaluated and some replaced with *true* or $false$, we can work up the model tree applying the operators to each leaf and achieve some simplification.

Pruning Algorithm

The pruning of a model to a box in Extended CSG, without applying the boundary or projection operator, can be described in algorithmic form as Algorithm 4.

The algorithm is recursive, starting at the root of the model and working up to the leaves. The evaluation is deferred by calling $ExtPrune()$ again if the node is still an operator. If the node is not one of our Extended CSG operators, we have reached the leaf and procedure $Eval()$ is called to evaluate that particular

formula to the box.

Note that, as in the case of CSG operator, after the pruning process, (M', B) should define the same set as (M, B).

Algorithm 4 $ExtPrune(M, B)$

Input: M, B
Output: M', B

```
if (M = Boundary{M}) then
  (M', B) ← ExtPrune(M, B)
  if (M' = true) then
    return(false, B)
  else if (M' = false) then
    return(false, B)
  else
    return(Boundary(M'), B)
else if (M = Projection_{V}{M}) then
  (M', B) ← ExtPrune(M, B)
  if (M' = true) then
    return(true, B)
  else if (M' = false) then
    return(false, B)
  else
    return(Projection_{V}{M'}, B)
else if (M = union{M_1, ..., M_k}) then
  ... as in Prune(M, B)
else if (M = intersection{M_1, ..., M_k}) then
  ... as in Prune(M, B)
else if (M = complement{M_1}) then
  ... as in Prune(M, B)
else
  return(Eval(M, B))
```

6.3.3 Recursive Subdivision

The main purpose of the subdivision of boxes is to allow each model to be simplified over a smaller box. Thus the model has more chance to be simplified. Note that the union of all sub-boxes with their models is supposed to define the same set as the original box with the original model.

As in the case of CSG, the subdivision can be carried on recursively along with the pruning process until some conditions are met. The set definition of (M, B) is also recursive on the structure of M. That is, if $B = B_1 \cup B_2$ it follows that:

$$(M, B) = (M, B_1) \cup (M, B_2)$$

In other words, when a box is divided, the model is pruned to its own box and the union of all these boxes with their models results in the original box with the original model.

For example, given a box B and a model $M = Proj(M_1 \cap M_2) \cup Boundary(M_3)$ If the box is subdivided into two sub-boxes, B_1 and B_2 along one of its coordinate, then:

$$\begin{aligned}
(M, B) &= (Proj(M_1 \cap M_2) \cup Boundary(M_3), B) \\
&= (Proj((M_1, B) \cap (M_2, B), B) \cup Boundary(M_3, B), B) \\
&= (Proj((M_1, B_1) \cap (M_2, B_1), B_1) \cup Boundary(M_3, B_1), B_1) \cup \\
&\quad (Proj((M_1, B_2) \cap (M_2, B_2), B_2) \cup Boundary(M_3, B_2), B_2)
\end{aligned}$$

However, the two new operators introduced in Extended CSG need some special considerations.

Overlapping Boxes

Suppose there are some boundary operators in M. An unfortunate subdivision might lose some boundary points on the edge of the divided boxes.

For example, suppose the model $M = union(x \geq 0, x \leq 1)$ and the box $B = [-1, 1]$. In B there are some points below $\{1\}$ but there are no points in B above $\{1\}$. It follows that $\{1\}$ is not arbitrarily close to $\overline{(M, B)}$ hence $(Bnd(M), B) = Boundary(M, B) = \{0\}$. Suppose we split B into $B_1 = [-1, 0]$ and $B_2 = [0, 1]$. The set (M, B_1) is empty so $(Bnd(M), B_1)$ is empty. (M, B_2) is B but $(Bnd(M), B_2)$ is also empty. This happened just because there was a points on

the boundary at the place where the split was done.

That is, boxes need to be overlapping to ensure the correctness of the set defined by (M, B) after the subdivisions.

Projected Variables

Consider the case of a model M with the variable list (x, y). Let y be the variable to be projected out. $M = intersection(CylProj_y(M_1), CylProj_y(M_2))$ inside the box B defines the set A, an area above the line segment on x-axis (Figure 6-2).

If we subdivided B along y-axis then:

$$A = (intersection(CylProj_{\{y\}}(M_1), CylProj_{\{y\}}(M_2)), B)$$

$$\begin{aligned} B &= union((intersection(CylProj_{\{y\}}(M_1), CylProj_{\{y\}}(M_2)), B_1), \\ &\quad (intersection(CylProj_{\{y\}}(M_1), CylProj_{\{y\}}(M_2)), B_2)) \\ &= \phi \end{aligned}$$

We cannot always subdivide the side that correspond to the variables which is to be projected. There are cases where splitting the side would be correct. For example, if $M = union(CylProj_y(M_1), ClyProj_y(M_2)$ but we do not consider these possibilities now.

In any case, in the application we will consider, the box B_p which correspond to bound variables is small relative to the box B_t which correspond to free variables. In other words, the variable which are to be projected are bounded to a relatively small intervals. Consequently, we can subdivide the Omnispace down to relatively small boxes without subdividing sides which correspond to bound variables.

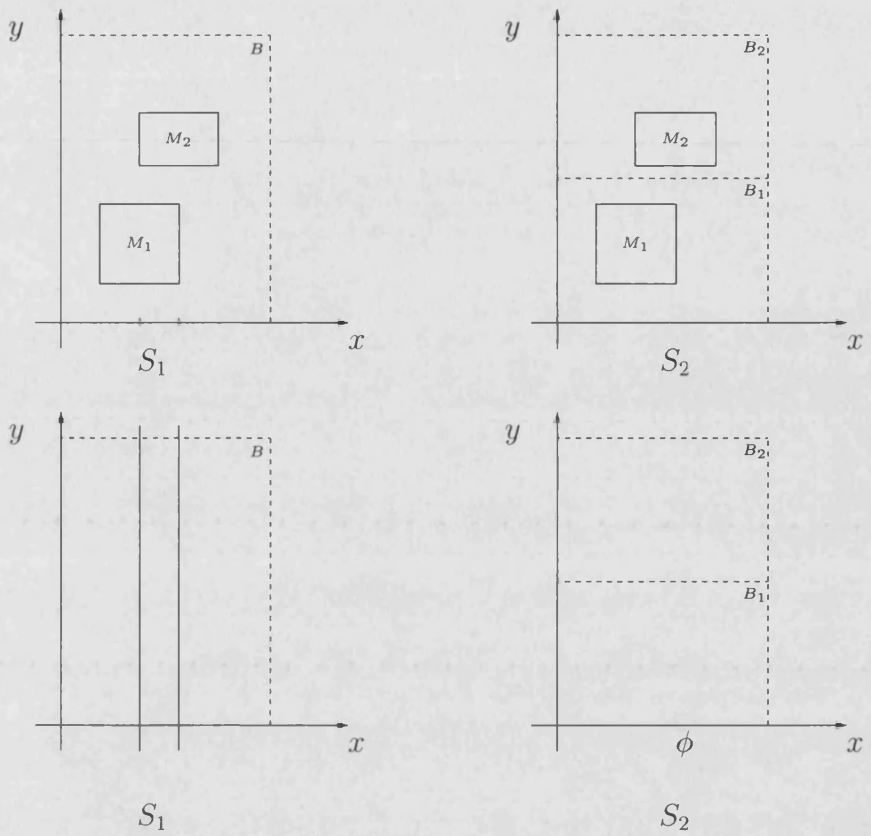


Figure 6-2: It is not always possible to subdivide the side that correspond to the variables which is to be projected.

Recursive Subdivision Algorithm

Given a box B and a model M, the subdivision process which divides the longest side of a box can be expressed in algorithmic form as in Algorithm 5. This process subdivides B into two sub-boxes, B_1 and B_2. The union of the sets defined by (M, B_1) and (M, B_2) is the same as the set defined by (M, B).

The process also look at special cases where M contains boundary operators. So by increasing one of the box by some small number $\epsilon > 0$, two new boxes are overlapping.

Also we cannot subdivide the side which correspond to the projected variables. By obtaining the set of free variable first, the longest side of the box can then be determined from the set.

Algorithm 5 $ExtSubDivide(M, B)$

Input: M, B

Output: $(M, B_1), (M, B_2)$

$freevar \leftarrow FindFreeVar(M)$
$position \leftarrow MaxSidePosition(B, freevar)$
$interval \leftarrow Part(B, position)$
$lower \leftarrow LowerEnd(interval)$
$upper \leftarrow UpperEnd(interval)$
$midpoint \leftarrow lower + Size(interval)/2$
$B_2 \leftarrow Part(B, position) \leftarrow [midpoint, upper]$
if $ContainBoundary(M) = true$) **then**
  $B_1 \leftarrow Part(B, position) \leftarrow [lower, midpoint]$
else
  $B_1 \leftarrow Part(B, position) \leftarrow [lower, midpoint + \epsilon]$
$return\ (M, B_1), (M, B_2)$

The division can be carried out recursively along with the pruning process until some conditions are met. A few options to consider as a termination condition of the process are, for example, the recursive subdivision could stop when all the boxes are sufficiently small or when the model is simple enough that is there are a certain number of primitives left in the model.

Algorithm 6 $ExtRecurSubDivision(M, B)$

Input: M, B

Output: $(M_1, B_1), \ldots, (M_k, B_k)$

$(M', B) \leftarrow Prune(M, B)$
if $IsSimple(M')$ *or* $IsSmall(B)$ **then**
  $return(M', B)$
else
  $(M, B_1), (M, B_2) \leftarrow ExtSubDivide(M, B)$
$return(ExtRecurSubDivision(M, B_1), ExtRecurSubDivision(M, B_1))$

6.3.4 Normal Forms

Semi-algebraic CSG obstacles can be expressed in Disjunctive Normal Form (DNF) or in Conjunctive Normal Form (CNF), or in some mixed form. Pruning and QE are both dependent on the input form. Recursive subdivision could also be considered as pre-processing for conversion from CNF to DNF, as well as for QE; or for conversion from a mixed form into either CNF or DNF.

Define a model to be in *normal form* (NF) if it is a union of projections of models with only boolean operators. A theorem due to Gabrielov [26] implies that any model over a box is equivalent to a normal form model over the same box. That is, for any (M, B) there exists $(NF(M), B)$ so that $NF(M)$ is in normal form and (M, B) defines the same set as $(NF(M), B)$.

The essential step in the proof is to show that the sets defined by normal form models are closed under compliment. Currently, we do not have any practical way to construct these normal form representations.

6.4 Extended CSG Approach to Spatial Planning

6.4.1 Representing C-space Obstacles

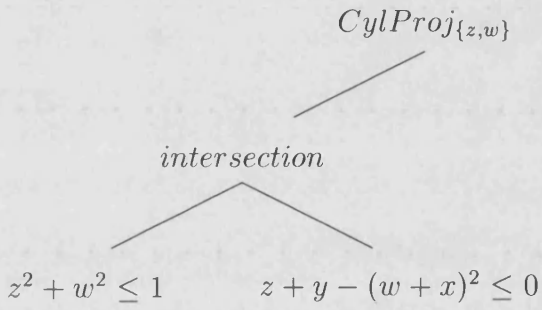
In Extended CSG, an object is represented by a tree with Boolean operators and also with extra operators, namely boundary and bounded projection. We cannot always replace these operators by Boolean ones. For example, it is only possible in principle, to replace the projection operator by Boolean operators applied to extend primitive if the variables which are to be projected out, occur algebraically in the atomic formulae.

However, we could work directly with the representation of the C-space obstacle as a projection in an extended CSG system. In any case, if the moving object or

part of the obstacle is not algebraic, we must represent the C-space obstacle as a projection, since elimination of quantifiers may not be possible.

Consider the following examples.

Example 4 *Let B be a box $([-10,10],[-10,10],[-10,10],[-10,10])$ in $zwxy$ space. Let M be a model define by:*



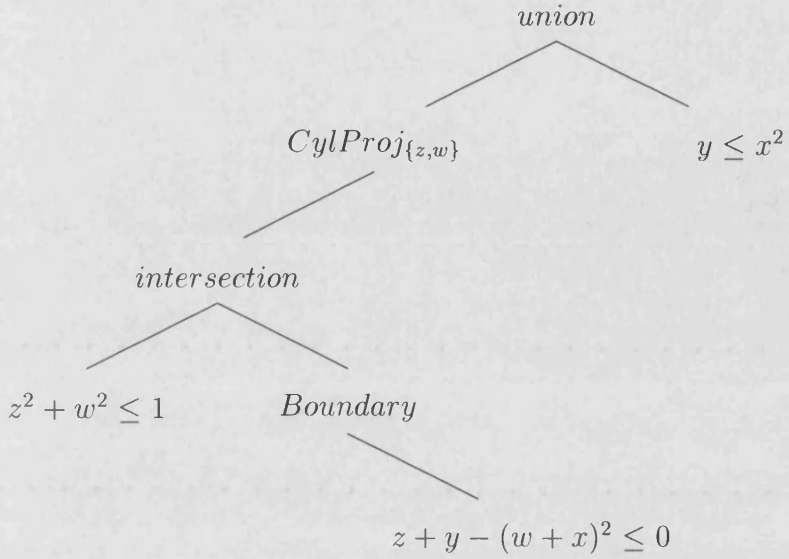
If we regard the unit circle $z^2 + w^2 \leq 1$ as an object in zw space which may translate and regard the area below the parabola defined by $w \leq z^2$ as an obstacle also in zw-space, then (M, B) represents the C-space obstacle in $zwxy$-space.

Using the cylindrical projection interpretation of projection operator, a point (z, w, x, y) is in (M, B) if it is in B and if there is a point (z', w') in the unit circle so that (z', w') translated by (x, y) lies below the parabola. In other words, the translation of the unit circle intersects the area below the parabola. Whether or not a point (z, w, x, y) is in (M, B), does not depend on z and w at all as long as they are in the projection of the box onto xy-space.

If we use the usual projection interpretation, (M, B) defines a set in xy-space inside the projection of B. It is the set of points (x, y) whose distance from the region below the parabola is no more than one unit.

We can also use the boundary operator to define the C-space obstacles as shown in Chapter 5.

Example 5 *Let B be a box $([-10,10],[-10,10],[-10,10],[-10,10])$ in $zwxy$ space. Let M be a model define by:*



Regard the unit circle $z^2 + w^2 \leq 1$ as an object which may translate and the area below a parabola defined by $y \leq x^2$ as an obstacle in xy-space. The circle is connected so it is always either entirely inside the obstacle or is intersected with at least an edge of the obstacle. That is, (M, B) represents the C-space obstacle.

In both examples, (M, B) can be defined as a semi-algebraic set without using projection or quantification. However, the process of finding the semi-algebraic definition, and the definition itself is complicated. It appears to be true that the most compact way to represent a set such as (M, B) is by using projection.

6.4.2 Partial Solutions to Find-space and Find-path Problems

Consider an Omnispace with an Omnimodel inside. After a cylindrical projection of the Omnimodel, although the Omnispace does not get reduced to a lower dimension C-space, the cylindrically projected Omnimodel is free of the variables

which were projected out. Therefore we can treat the Omnispace as the C-space by ignoring coordinates which correspond to projected variables.

The pruning of a model to a box in extended CSG can be done without applying the boundary or cylindrical projection operator. Suppose we apply only subdivisions and pruning to the Omnimodel over the Omnispace. Every sub-box which has no part of Omnimodel inside is not a part of the projection of the Omnimodel. Similarly, every sub-box which is totally inside the Omnimodel is also inside the projection of the Omnimodel.

Clearly, the result from this process is conservative. To obtain the exact value, we would need to evaluated boxes which are *undecided* after we apply boundary and projection operators.

Find-space

Since the Freespace is a collection of all points outside the C-space obstacles, we can also represent the Freespace in the same way that we represent C-space obstacles.

Suppose we have an Omnispace, we can represent the Freespace as an Omnimodel in this space. If we apply pruning and subdivisions so that the Omnispace is divided into sub-boxes of a certain size, then sub-boxes which are evaluated to *true* are in the Freespace. These sub-boxes are part of the Freespace thus they are solutions to Find-space problem. If there is no *true* box then, at this level of subdivisions, there is no Freespace.

We can describe this partial and conservative process to find Freespace in algorithmic form as in Algorithm 7.

For example, consider a convex object P defined by a combination of four atomic formulae of degree one as:

$$P = intersection(z \geq 0, z \leq 1, w \geq 0, w \leq 2).$$

Algorithm 7 $Findspace(M, B)$

Input: M, B

Output: $B_1, \ldots, B_k$

$boxlist = \{\}$
$(M', B) \leftarrow Prune(M, B)$
if $(M', B) = true$ **then**
  $return(B)$
else if $IsSmall(B)$ **then**
  $return()$
else
  $(M, B_1), (M, B_2) \leftarrow ExtSubDivide(M, B)$
$return(ExtRecurSubDivision(M, B_1), ExtRecurSubDivision(M, B_1))$

Also consider a set of 15 convex objects O, each defined by a combination of four atomic formulae of degree one as:

$$
\begin{aligned}
O \;=\; & union(\\
& intersection(z \geq 0, z \leq 1, w \geq 0, w \leq 1), \\
& intersection(z \geq 4, z \leq 5, w \geq 3, w \leq 4), \\
& intersection(z \geq 8, z \leq 9, w \geq 12, w \leq 13), \\
& intersection(z \geq 16, z \leq 17, w \geq 6, w \leq 7), \\
& intersection(z \geq 20, z \leq 21, w \geq 5, w \leq 6), \\
& intersection(z \geq 24, z \leq 25, w \geq 4, w \leq 5), \\
& intersection(z \geq 28, z \leq 29, w \geq 7, w \leq 8), \\
& intersection(z \geq 32, z \leq 33, w \geq 8, w \leq 9), \\
& intersection(z \geq 36, z \leq 37, w \geq 2, w \leq 3), \\
& intersection(z \geq 40, z \leq 41, w \geq 1, w \leq 2), \\
& intersection(z \geq 44, z \leq 45, w \geq 10, w \leq 11), \\
& intersection(z \geq 48, z \leq 49, w \geq 14, w \leq 15), \\
& intersection(z \geq 52, z \leq 53, w \geq 13, w \leq 14), \\
& intersection(z \geq 56, z \leq 57, w \geq 11, w \leq 12), \\
& intersection(z \geq 60, z \leq 61, w \geq 9, w \leq 10)).
\end{aligned}
$$

Let (z, w, x, y, a) be the variable list available for both the object and obstacles and $[0,1] \times [0,1] \times [0,64] \times [0,16] \times [0,7]$ be the box. After the pruning and subdivision which apply to only boxes corresponding to variables x and y using the maximum box size of 4, we obtain 64 sub-boxes in which 20 were in the Freespace (Figure 6-3).

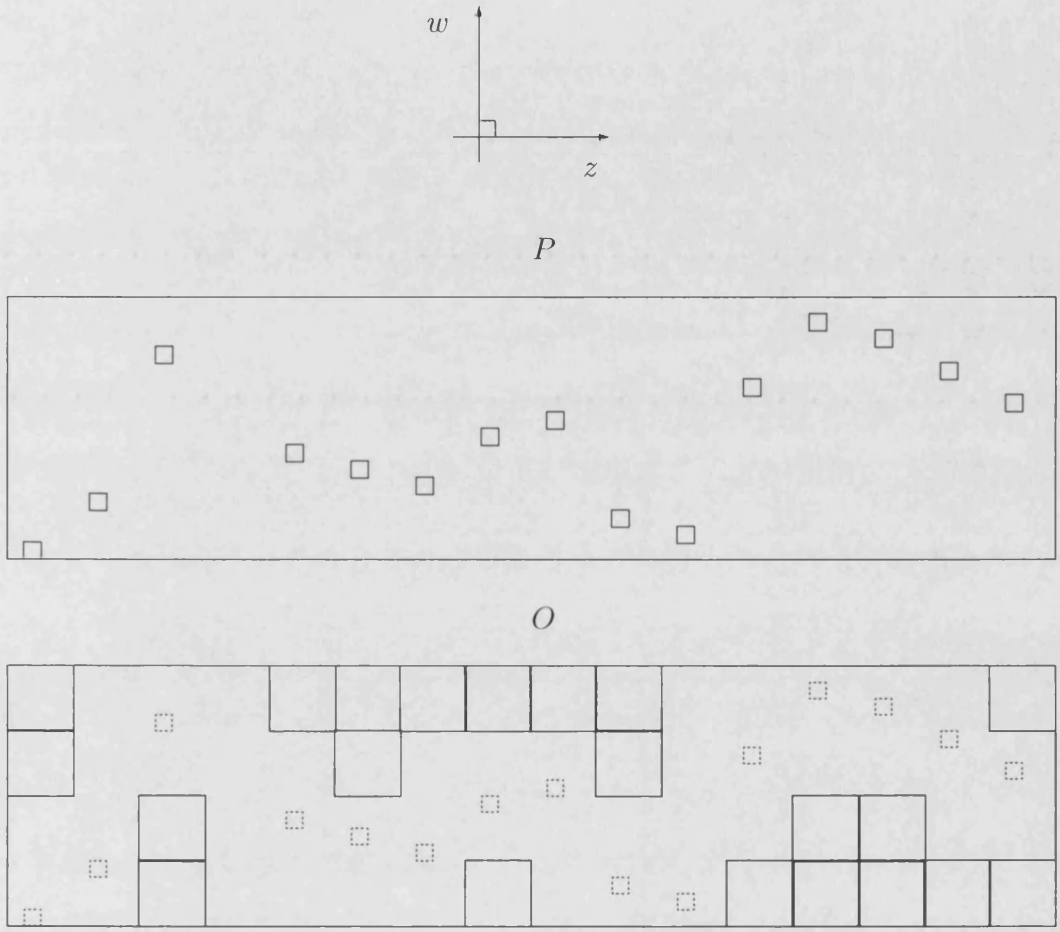


Figure 6-3: An object, the set of obstacles and the partial Freespace.

Additionally, we subdivide the box on sides which correspond to variables x, y and a using the maximum box size of 4. After the pruning we obtain 128 sub-boxes in which 45 were in the Freespace (Figure 6-4).

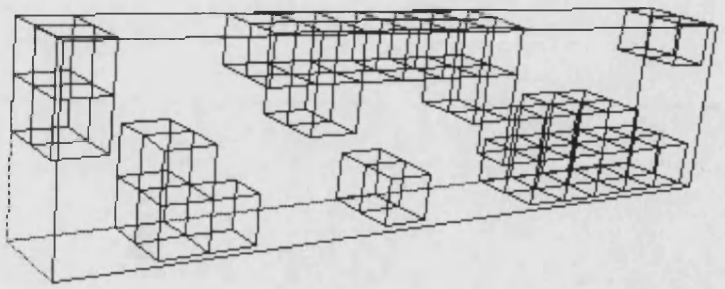


Figure 6-4: Partial Freespace in 3-dimensions.

Find-path

Find-path problem concerns with moving an object from one place to another without collisions, hence a path within the Freespace is a solution.

After applying the process of pruning and subdivision to the Omnispace we may get a list of sub-boxes which correspond to Freespace. We can represent these sub-boxes using an adjacency matrix. Once we have the adjacency matrix, Find-path problem can be solved using standard graph searching techniques.

Denote these sub-boxes by $B_1, \ldots, B_k$. For a box B_i and B_j in the Omnispace $B_p \times B_t \times B_r$, define B_i and B_j to be connected if their edges in B_t and B_r are overlapping. The adjacency matrix A for k sub-boxes is a $k \times k$ matrix $[a_{ij}]$

$$a_{ij} = \begin{cases} 1, & \text{if } B_i \text{ connected to } B_j, \\ 0, & \text{if } B_i \text{ not connected to } B_j. \end{cases}$$

However, B_r represent the degree of rotations hence the edge 0 is overlapping with the edge 2π radians.

6.5 Summary

We introduced an extended version of semi-algebraic CSG which has projection and boundary as operators. We then showed how this idea can be applied to spatial planning by representing C-space obstacle using primitives from this system. In particular, we also suggest how to partially solve Findspace and Findpath problem using this system.

Chapter 7

Conclusions and Further Work

We are interested in spatial planning problems. In particular, we are interested in computing constraints on the position and orientation of a 2-dimensional object due to the presence of some obstacles.

We were first inspired by the concept of the Configuration Space (C-space) approach to spatial planning. The C-space approach reduces the problem from having to deal with the intersections between a set of objects in the Workspace to dealing with a point relative to a set of objects instead. This new set of objects, called the C-space obstacle, corresponds to all the impossible positions and orientations of the actual object in the Workspace. However, the dimension of C-space obstacles is at least the same as the degrees of freedom of the object in the Workspace.

To construct the C-space obstacles we employed the modelling technique used by Svlis geometric modeller. Svlis represents object using the concept of Constructive Solid Geometry (CSG) and extended semi-algebraic sets which are dimensionally independent. Moreover, the representation of objects in Svlis allows processes such as pruning and recursive subdivisions which are 'divide and conquer'-type of simplification.

The second inspiration came from the quantifier elimination technique called

'cylindrical algebraic decomposition'. It was shown that in principle this quantifier elimination technique could be used to solve spatial planning problem which was represented in semi-algebraic forms.

Our first contribution was to suggest *constructing C-space obstacles in semi-algebraic form using existential quantifiers and boundary formations* (see Equation 5.2 and Equation 5.5, Section 5.1.1). This quantified representation of C-space obstacle is equivalent to the concept of the Omnimodel which was independently and simultaneously studied by Wise in [68].

In order to represent the C-space in CSG with Boolean operators alone, one approach is to apply elimination of quantifiers. However, elimination of quantifiers has computational difficulty which increases much more than linearly with complexity of formula. Therefore, we introduce another original idea of *using spatial subdivision techniques as a pre-process before applying quantifiers elimination.*

We implemented the idea in REDUCE computer algebra system using REDLOG logic package. Our computational results showed only slight benefit from pruning and subdivision process with large and incomprehensible output.

We concluded that, although possible in theory, using cylindrical algebraic decomposition to elimination quantifiers, is computationally hard and the quantifier-free results are often large and cumbersome. Therefore, we introduce another original idea of *constructing an extended Constructive Solid Geometry system* which would have both bounded projection and boundary formation as operators, as well as the usual Boolean ones.

Our application of Extended CSG to spatial planning leads to a compact representation of configuration space obstacles. In particular, it allows the computation of partial solutions to Find-space and Find-path problems without applying quantifier elimination. Since the pruning and recursive subdivision can be done on an extended CSG representation, there is also a possibility to apply quantifier elimination to limited regions efficiently.

It seems that we should learn to work directly with the representation of the

C-space obstacle as a projection in an extended CSG system. In any case, if the moving object or part of the obstacle is not algebraic, we must represent the C-space obstacle as a projection, since elimination of quantifiers may not be possible.

Although the extended CSG system, at present, does not provide a practical method which solve the problem in all cases, it does, at least, provide a compact semi-algebraic CSG representation.

7.1 Implication for Further Research

The current quantifier elimination software used in this thesis is somewhat limited. However, quantifier elimination by cylindrical algebraic decomposition is difficult and there will always be a limitation on the complexity of the formulae. Additionally, the algebraic description of the C-space obstacle, or the Freespace could be obtainable with a better quantifier elimination software but it maybe too complicated thus making it unusable.

Since we use interval arithmetic conservatively, the more precise interval arithmetic would enable a more accurate pruning process. This will have an implication on pruning and subdivisions when used as a pre-process to quantifier elimination. Find-path and Find-space using only pruning and subdivision would also benefit from this improvement.

Another consideration is to subdivide the moving object as well as the obstacles. Furthermore, the boxes which correspond to the bound variables can also be divided. However, the benefit of these approaches are still unclear.

Finally, there maybe other ideas which could be combined with other techniques to solve spatial planning problem with more efficiency and precision.

Appendix A

REDUCE Procedures

A.1 Grid Division Procedure

```
% Quantifier Elimination using REDLOG.
%----------
procedure apply_qe(form,nobv,v_list);
% apply qe and return a quantifier free formula
% v_list -- bound variable first
begin
    scalar i,bv_list,non_quantified;
    write "Total number of terms: ",rlatnum(form);
    if (form = true) or (form = false) then
        non_quantified := form
    else
  begin
        write "Apply QE to ",form, " ... ";
        bv_list := for i := 1: nobv collect part(v_list,i);
        non_quantified := rlqe(ex(bv_list,form));
    end;
```

```
    return (rlsimpl(non_quantified));
end;

%----------
procedure QE_Grid(piano,obstacle,box,nobv,v_list,max_size);
% Grid subdivision
% - Chop 'til all sides of each box <= max_size
%   then put all the boxes into a list
% - apply_qe to pruned boxes.
%
% - list_box is a list of boxes,
% - piano in {z,w}
% - obstacle in {z,w}
begin
    scalar j,list_box,bux,result;
    list_box := chop_freevar_to_size(box,nobv,max_size);
    result := for each bux in list_box mkor
        apply_qe(
            tree2form(
                {and,
                 box2md1(bux,length(v_list),v_list),
                 prune({and,piano,w2c(obstacle,nobv,v_list)},
        bux,
          v_list)},
    nobv,
                v_list),
 nobv,
 v_list);
  write "Number of box(es) : ", length(list_box);
  write list_box;
  return(result);
end;
```

A.2 Recursive Subdivision Procedure

```
%----------
procedure QE_Recur(piano,obstacle,box,nobv,v_list,max_size);
% recursive subdivision
begin
    scalar model;
    model := {and,
              piano,
              w2c(obstacle,nobv,v_list)};
 result := recur_sub(model,box,nobv,v_list,max_size);
 return(result);
end;

%----------
procedure recur_sub(model,box,nobv,v_list,max_size);
% recursive subdivision
begin
    scalar l,fv_box,box1,box2,result,pruned_model,value;
    l := length(box);
    fv_box := for j:=(nobv+1):l collect part(box,j);
    pruned_model := prune(model,box,v_list);
    value := rlsimpl(tree2form(pruned_model,nobv,v_list));
    if (value = true) then
    begin
        write box," is solid.";
        result :=
        apply_qe(
            tree2form(
                box2md1(box,length(v_list),v_list),
                nobv,
                v_list),
            nobv,
            v_list);
```

```
    end
    else if (value = false) then
    begin
        write box, " is empty [",max_side_size(fv_box),"].";
        result := false;
    end
    else if (max_side_size(fv_box) < max_size ) then
    begin
        write "Prune and QE ", box;
        result :=
            apply_qe(
                tree2form(
                    {and,
                     box2md1(box,length(v_list),v_list),
                     pruned_model},
                    nobv,
                    v_list),
            nobv,
            v_list);
    end
    else
    begin
        box1 := part(chop_freevar(box,nobv,v_list),1);
        box2 := part(chop_freevar(box,nobv,v_list),2);
        result :=
            recur_sub(pruned_model,box1,nobv,v_list,max_size) or
            recur_sub(pruned_model,box2,nobv,v_list,max_size)
    end;
    return(result);
end;

%----------
```

References

[1] B. Aronov and M Sharir. On Translational Motion Planning of a Convex Polyhedron in 3-Space. *Society for Industrial and Applied Mathematics*, 26(6):1785–1803, 1997.

[2] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical Potential Field Techniques for Robot Path Planning. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):224–241, 1992.

[3] J. Barraquand and J. C. Latombe. Robot Motion Planning: a Distributed Representation Approach. *International Journal of Robotics Research*, 10(6):628, 1991.

[4] Riccardo Benedetti and Jean-Jacques Risler. *Real Algebraic and Semi-Algebraic sets.* Hermann, 1990.

[5] A. Bicchi, G. Casalino, and C. Santilli. Planning Shortest Bounded-Curvature Paths for a Class of Nonholonomic Vehicles Among Obstacles. *Journal of Intelligent & Robotic Systems*, 16(4):387–405, 1996.

[6] O. Bottema and B. Roth. *Theoretical Kinematics.* North-Holland Publishing Company, Amsterdam, 1979.

[7] A. Bowyer. *Svlis—Introduction and User Manual.* Information Geometers, 1995.

[8] M. S. Branicky and W. S. Newman. Rapid Computation of Configuration Space Obstacles. *Proc 1990 IEEE International Conference Robotics Automation*, pages 304–310, 1990.

[9] R. A. Brooks. Solving the Find-Path Problem By Good Representation of Free Space. *IEEE Transactions on Systems Man and Cybernetics*, 13(Mar/Apr):190, 1983.

[10] R. A. Brooks and T. Lozano-Pérez. A Subdivision Algorithm in Configuration Space for Findpath with Rotation. *IEEE Transactions on Systems Man and Cybernetics*, 15(2):224–233, 1985.

[11] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, Mass., 1988.

[12] Y. P. Chien and Q. Xue. Path Planning for Two Planar Robots Moving in Unknown Environment. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):307–317, 1992.

[13] G. E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings of the 2nd GI Conference*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183. Springer, Berlin, 1975.

[14] J. H. Davenport. On a 'Piano Movers' Problem. *ACM SIGSAM Bulletin*, pages 15–17, 1986.

[15] J. H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra: Systems and Algorithms for Algebraic Computation.* Academic Press, 1988.

[16] Davenport, J. H. and Heintz, J. Real Quantifier Elimination is Doubly Exponential. *Journal of Symbolic Computation*, 5(1-2):29–36, February/April 1988.

[17] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 1997.

[18] G. Desaulniers and F. Soumis. An Efficient Algorithm to Find a Shortest-Path for a Car-Like Robot. *IEEE Transactions on Robotics and Automation*, 11(6):819–828, 1995.

[19] D. Dobkin, L. Guibas, J. Hershberger, and J. Snoeyink. An Efficient Algorithm for Finding the CSG Representation of a Simple Polygon. Technical report, DEC System Research Center, September 1989.

[20] A. Dolzmann and T. Sturm. REDLOG User Manual. Technical Report MIP-9616, Fakultät für Mathematik und Informatik, Universität Passau, October 1996. for REDLOG Version 1.0.

[21] A. Dolzmann and T. Sturm. REDLOG: Computer Algebra Meets Computer Logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.

[22] A. Dolzmann and T. Sturm. REDLOG User Manual. Technical Report MIP-9905, Fakultät für Mathematik und Informatik, Universität Passau, April 1999. for REDLOG Version 2.0.

[23] B. Faverjon. Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator. In *Proceedings of the IEEE Int. Conf. Robotics, Atlanta, GA*, pages 504–512, March 1984.

[24] K. Fujimura. Motion Planning Amidst Dynamic Obstacles in Three 1 Dimensions. *1993 International Conference on Intelligent Robots and Systems*, page 1387, 1993.

[25] K. Fujimura and H. Samet. Motion Planning in a Dynamic Domain. *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 324–330, 1990.

[26] A. Gabrielov. On Complements of Subanalytic Sets and Existential Formulas for Analytic Functions. *Invent. Math.*, 125:1–12, 1996.

[27] A. J. Gomes and J. C. Teixeira. A mathematical framework for set-theoretic solid models. In *CSG 94 Set-theoretic Solid Modelling: Techniques and Applications*, pages 19–33. Information Geometers, 1994.

[28] A. C. Hearn. *REDUCE User's Manual Version 3.7.* RAND Publication CP78, 1999.

[29] H. Hong. *Improvements in CAD-based Quantifier Elimination.* PhD thesis, Ohio State University, 1990.

[30] H. Hong. Approximate Quantifier Elimination. In *Proceedings of SCAN'95*, 1995. Invited Talk.

[31] J. E. Hopcroft, J. K. Kearney, and D. B. Krafft. Case Study of Flexible Object Manipulation. *International Journal of Robotics Research*, 10(1):41–50, 1991.

[32] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the Complexity of Motion Planning for Multiple Independent Objects: P-space-hardness of the "Warehouseman's Problem". *International Journal of Robotics Research*, 3(4):76, 1984.

[33] J. E. Hopcroft and G. T. Wilfong. Reducing Multiple Object Motion Planning To Graph Searching. *SIAM Journal on Computing*, 15(3):768–785, 1986.

[34] Y. K. Hwang and N. Ahuja. A Potential-Field Approach To Path Planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32, 1992.

[35] Y. K. Hwang and N. Ahuja. Gross Motion Planning — a Survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

[36] M. Kalkbrener and S. Stifter. Some Examples for Using Quantifier Elimination for the Collision Problem. Technical Report 87-22, Research Institute for Symbolic Computation (RISC), Linz., September 1987.

[37] Myung-Soo. Kim, Jae-Woo. Ahn, and Soon-Bum. Lim. An Algebraic Algorithm to Compute the Exact General Sweep Boundary of a 2D Curved Object. *Information Processing Letters*, 47(5):221–229, 1993.

[38] Myung-Soo Kim and K. W. Nam. Interpolating Solid Orientations with Circular Blending Quaternion Curves. *Computer-Aided Design*, 27(5):385–398, 1995.

[39] Y. Koren. *Robotics for Engineers.* McGraw Hill, 1985.

[40] J. Latombe. *Robot Motion Planning.* Kluwer Academic Publishing, 1993.

[41] T. Lozano-Pérez. Spatial Planning: a Configuration Space Approach. *IEEE Transactions on Computers*, 32(2):108–119, 1983.

[42] V. J. Lumelsky. Dynamic Path Planning for a Planar Articulated Robot Arm Moving Amidst Unknown Obstacles. *Automatica*, 23(5):551–570, 1987.

[43] V. J. Lumelsky. Effect of Kinematics on Motion Planning for Planar Robot Arms Moving Amidst Unknown Obstacles. *IEEE Journal of Robotics and Automation*, 3(3):207–223, 1987.

[44] M. A. H. MacCallum and F. J. Wright. *Algebraic Computing with REDUCE.* Oxford University Press, 1991.

[45] S. McCallum. Partial Solution to a Path Finding Problem Using the CAD Method. Technical report, Department of Computing, Macquarie University, Australia, June 1995.

[46] E. R. Moore. *Interval Analysis.* Prentice-Hall, Inc., 1966.

[47] T. J. Pan and R. C. Luo. Motion Planning for Mobile Robots in a Dynamic Environment with Moving Obstacles. *Proc 1990 IEEE International Conference Robotics Automation*, 1:578, 1990.

[48] S. Parry-Barwick and A. Bowyer. Multidimensional set-theoretic feature recognition. *Computer-Aided Design*, 27(10):731–740, 1995.

[49] S. J. Parry-Barwick. *Multi-dimensional Set-theoretic Geometric Modelling.* PhD thesis, Dept. of Mechanical Engineering, University of Bath, 1995.

[50] S. J. Parry-Barwick and A. Bowyer. Woodwark's Method for Feature Recognition. *Technical Report 099/1992*, 1992.

[51] S. Ratschan. Approximate Qunatified Constraint Solving By Cylindrical Box Decomposition. Technical Report 00-27, Research Institute for Symbolic Computation (RISC), Linz., September 2000.

[52] S. Ratschan. Convergence of Approximate Constraint Solving by Approximate Quantifiers. Technical Report 00-23, Research Institute for Symbolic Computation (RISC), Linz., June 2000.

[53] S. Ratschan. Uncertainty Propagation in Heterogenous Algebras for Approximate Quantified Constraint Solving. Technical Report 00-23, Research Institute for Symbolic Computation (RISC), Linz., September 2000.

[54] J. H. Reif. Complexity of the Mover's Problem and Generalizations. *Proceedings of the 20th Symposium on the Foundations of Computer Science*, pages 421–427, 1979.

[55] J. H. Reif and M. Sharir. Motion Planning in the Presence of Moving Obstacles. *Journal Of The Association For Computing Machinery*, 41(4):764–790, 1994.

[56] J Renegar. Recent Progress on the Complexity of the Decision Problem for the Reals. In B. F. Caviness and J. R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 220–241. Springer-Verlag/Wien, 1998.

[57] J. T. Schwartz and M. Sharir. On the "piano movers" problem I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Comm. on Pure and Applied Math.*, 26:345–398, 1983.

[58] J. T. Schwartz and M. Sharir. On the "piano movers" problem II. general techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4:298–351, 1983.

[59] J. T. Schwartz and M. Sharir. On the piano movers' problem III: Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *International Journal of Robotics Research*, 2(3):46, 1983.

[60] J. T. Schwartz and M. Sharir. On the piano movers' problem V: The case of a rod moving in three-dimensional space amidst polyhedral obstacles. *Comm of Pure & Applied Maths*, 37:815, 1984.

[61] M. Sharir and E. A. Sheffi. On the piano movers' problem IV: Various decomposable-dimensional motion-planning problems. *Comm on Pure & Applied Maths*, 37:479, 1984.

[62] T. Skewis and V. Lumelsky. Experiments with a Mobile Robot Operating in a Cluttered Unknown Environment. *Journal of Robotic Systems*, 11(4):281–300, 1994.

[63] T. Sturm and V. Weispfenning. Computational Geometry Problems in REDLOG. Technical Report MIP-9708, Fakultät für Mathematik und Informatik, Universitä t Passau, April 1997.

[64] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California, Berkeley, 1951.

[65] N. Tongsiri and D. Richardson. Semi-algebraic Approach to Piano Movers' Problem. In X-S Gao and D. Wang, editors, *Computer Mathematics, Proceedings of the Fourth Asian Symposium on Computer Mathematics*, volume 8 of *Lecture Notes Series on Computing.* World Scientific Publishing, December 2000.

[66] A. F. Vanderstappen, D. Halperin, and M. H. Overmars. The Complexity of the Free-space for a Robot Moving Amidst Fat Obstacles. *Computational Geometry — Theory and Applications*, 3(6):353–373, 1993.

[67] K. D. Wise. Using multidimensional CSG models to map where objects can and cannot go. Technical Report 001/96, Dept. of Mechanical Engineering, University of Bath, January 1996.

[68] K. D. Wise. *Computing Global Configuration-Space Maps Using Multidimensional Set-Theoretic Modelling.* PhD thesis, University of Bath, 1999.

[69] J. Woodwark. *Geometric Reasoning.* Clarendon Press, Oxford, 1989.